NEW cover 4/18/01 1:03 PM Page 2

NANOMOTION™ II USER MANUAL

MELLES GRIOT

Table of Contents

Chapter 1

General Introduction

1.1 Copyright and Manual Notice

This manual describes the operation of the Nanomotion II equipment distributed by Melles Griot. Melles Griot and the manufacturer reserve the right to make changes to this manual and to the equipment described herein without notice. Melles Griot has made considerable efforts to ensure that the information in this manual is accurate and complete. However, Melles Griot will not be liable for any technical or editorial errors or omissions made herein or incidental, special, or consequential damages of any nature that result from the use of this manual, or operation and performance of equipment in connection with this manual.

1.2 Warranty Statement

The manufacturer warrants the Nanomotion II system against defects in materials and workmanship for a period of one (1) year from the date of purchase.

1.2.1 Life Support Policy

Nanomotion II is not authorized for use as a critical component in life support devices or systems without the prior written approval of the manufacturer. As used herein:

1. Life support devices or systems are devices or systems that (a) are intended for surgical implant into the body or (b) support or sustain life, and whose failure to perform, when properly used in accordance

with instructions for use provided in the labeling and associated literature, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

1.3 Manufacturer Statement

Nanomotion II is manufactured by:

Applied Precision, Inc.
1040 12th Avenue NW
Issaquah, WA 98027

Nanomotion II is a registered trademark of Applied Precision, Inc.

1.4 How To Use The Manual

1.4.1 Overview

This manual is a guide to the operation and use of the Nanomotion II micropositioning system in a user-friendly and readable format.

The manual will provide all of the information that needed to understand and operate the instrument. In Chapter 2, Introduction to Nanomotion II, the background and capabilities of the instrument are introduced, including concepts important to understanding the instrument functionality. Chapter 3, Specifications, describes the detailed performance parameters of the system and identifies the controls, indicators, and connectors on the system controller. Chapter 4, Installation and Setup describes how to configure the Nanomotion II system for operation, and how to install the Nanomover motors. Chapter 5, Operating the Nanomotion II System, shows how to operate the controller using the various software programs included with the system, and Chapter 6, Programming Nanomotion II, gives a complete description of commands and syntax provided with the system. Chapter 7, Application Notes, provides information about using the system in a real-world environment, and finally, appendices provide useful reference material.

For best results, read the entire manual before attempting to use Nanomotion II. At a minimum, first-time users should read the following chapters:

- Chapter 2: *Introduction to Nanomotion II*
- Chapter 4: *Installation and Setup*
- Chapter 5: *Operating Nanomotion II*

1.4.2 Conventions Used in the Manual

In order to help you understand the material presented in this manual, several text formats are used to highlight important information. The following examples demonstrate these text-formatting conventions.

Example #1

> **WARNING: Warnings are intended to alert the user to an important consideration about the operation or safe use of Nanomotion II.**

Example #2

References to other parts of the manual, such as *Operating Nanomotion II*, appear as italicized text.

Example #3

NOTE:

Notes are intended to further explain or offer exceptions to the text.

Example #4

Commands that you must type into the computer appear in a different font.

```
cd \LOCATION\windows
NanoMotn
```

Example #5

Command parameters that are optional are surrounded by square brackets.

```
variable [parameter1][parameter2]
```

Example #6

Computer functions, libraries, and related terms are indicated by a different font.

`nl_read_position`

1.5 Continuing Improvement

Melles Griot is dedicated to supporting Nanomotion II technology. Care has been taken to ensure that Nanomotion II is straightforward to use and is complete in all aspects. Please report errors and problems with Nanomotion II to Melles Griot.

Melles Griot strives to provide the customer with a high-quality, high-value product. Because we want to ensure the continuing improvement of our products, we encourage you, the user of Nanomotion II, to contact us with issues that you feel would improve the system. With your help, we can continue to produce products of the highest quality, value, and usefulness.

1.6 Customer Service

Contact Melles Griot for information about incorporating Nanomotion II into your original equipment (OEM) application or laboratory experiment.

Customer Service
Toll Free: (800) 835-2626
Phone: (949) 261-5600
Fax: (949) 261-7790
E-Mail: sales@irvine.mellesgriot.com

Chapter 2

Introduction to Nanomotion™ II

Congratulations on purchasing Nanomotion II, one of the finest nanometric positioning systems available today. This system makes possible computer-controlled motorized movements of very high resolution with superior repeatability and accuracy. This chapter of the manual introduces Nanomotion II and describes the capabilities of the instrument.

2.1 Overall Concept

Nanomotion II provides higher speed, better repeatability, and finer resolution than any other positioner in its class. With Nanomotion II, an operator can execute sophisticated moves with extreme precision via a user-friendly MS Windows interface. Alternatively, a custom application can be created to control the instrument using a high-level computer language.

The system consists of three components that function together — application software, controller chassis, and motorized positioner(s) — which, along with a user-supplied host computer, provide an easy-to-use, yet powerful, means for precise nanopositioning; To execute a movement, the user issues a command. The computer communicates these commands to the controller chassis, which contains electronics that examine the commands and apply power to the positioner motor(s). The motor moves in response, causing the micrometer screw to turn resulting in the desired linear motion.

2.2 History of Nanomotion

Nanomotion II was released in 1995 as a successor to the highly successful Nanomotion micropositioning system. While externally very similar to its predecessor, Nanomotion II is significantly improved. It offers better resolution (10 nm instead of 50 nm), an updated, MS Windows-based control application, a dynamic link library (DLL) for Windows programming, an IEEE and RS232 interface for advanced users, and LabVIEW drivers. With Nanomotion II, Melles Griot and the manufacturer continue to set the standard for mechanized nanometric positioning for the next generation of demanding applications.

2.3 Nanomotion™ II Configurations

There are three basic configurations of the Nanomotion II controller — the 11 NCS 101/IBM controller, designed to operate with an IBM-compatible personal computer using a PC-Link card mounted in the host computer; the 11 NCS 101/IEEE controller, intended for use with an IEEE 488 general purpose interface bus (GPIB) or an RS-232C serial interface; and the 11 NCS 101 expansion controller, used in conjunction with the 11 NCS 101/IBM or 11 NCS 101/IEEE when it is necessary to control more than two Nanomover motors. The specific configuration accompanying this manual is indicated on the inside front cover.

2.3.1 Controlling from an IBM-Compatible PC

If an IBM-compatible personal computer is used, there are two possible hardware configurations. In the first, a PC-Link board is inserted in the computer and a PC-interface board placed in the controller chassis (11 NCS 101/IBM). This system can then be operated with the Windows Control Program or a custom application that utilizes the Windows dynamic link library (DLL) or links to the MS-DOS C libraries. All of the control choices are available — using one does not preclude the other (see Figure 2.1).

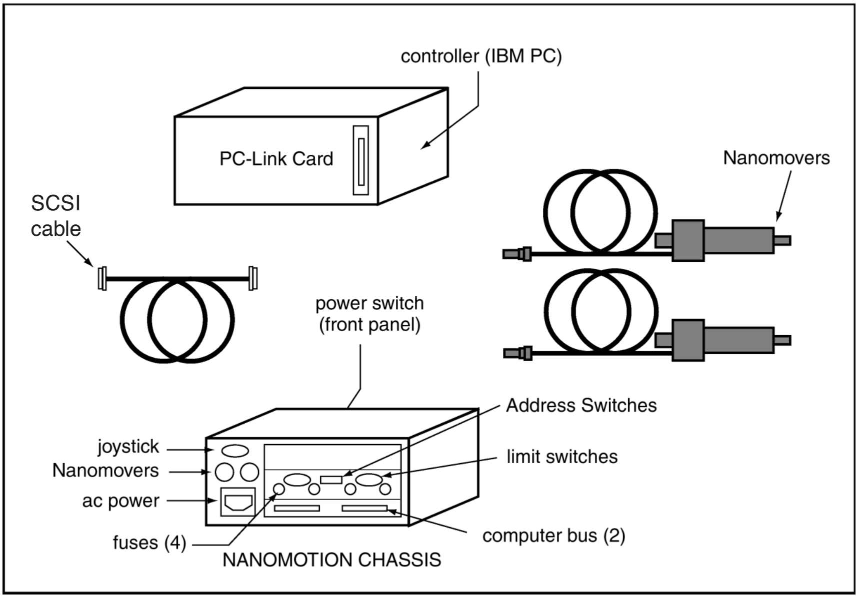


Figure 2.1 IBM System

In the second PC configuration, an IEEE 488.2/RS-232C interface board is inserted into the controller chassis, and no additional hardware, other than a user-provided cable, is added to the PC. The system can then be operated with LabVIEW, using either the provided drivers or the IEEE 488.2/RS-232C command set, or with a user-written program that controls the chosen interface port (see Figure 2.2).

2.3.2 Controlling from an RS-232C or IEEE 488.2 Interface

If a non-IBM-compatible computer is selected, the IEEE 488.2/RS-232C interface board must be used in the controller chassis. Either the IEEE 488.2 or the RS-232C interface capability of the board can be utilized according to the preferences of the user (see Figure 2.2).

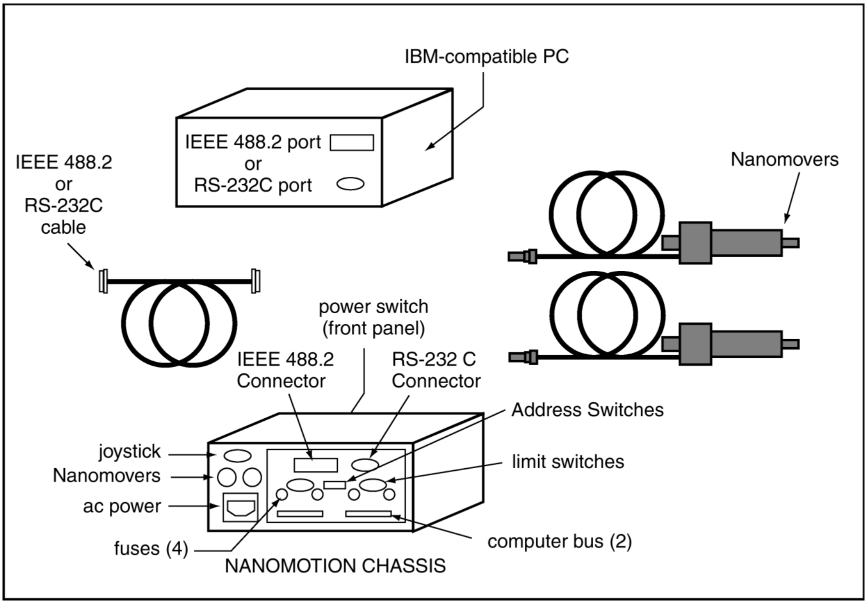


Figure 2.2 IEEE System showing optional expansion port in lower bay

2.3.3 Controlling more than two Nanomover motors

In all of the configurations described above, up to seven additional controller chassis can be added to the base system. With the full complement eight chassis, a total of 16 Nanomover motors (two per chassis) can be controlled by a single host computer. The controller chassis are connected in series through the PC-interface boards that must be present in each chassis.

2.4 System Components

2.4.1 Nanomotion II PC-Link Board

[Included with the 11 NCS 101/IBM Controller] Installed in a host personal computer, the Nanomotion II PC-Link interface board provides the connection between an IBM PC (AT or later) and a Nanomotion II controller chassis. Essentially, the PC-Link board is a PC bus extender, which provides all of the digital signals that the Nanomotion II controller chassis requires.

2.4.2 Nanomotion II Controller Chassis

The controller chassis contains electronics to control and operate one or two Nanomovers. Each controller chassis contains a power supply, a two-channel Nanomover control board, and two card bays, one for an IBM ISA interface board and one for an IEEE 488.2/RS-232C interface card. Up to eight controller chassis can be connected together, allowing for up to a total of 16 Nanomovers to be controlled from the same host computer. These configurations are described in Chapter 4, Installation and Setup.

2.4.3 IEEE 488.2/RS-232C Interface Card

[Included with the 11 NCS 101/IEEE Controller] This card fits in the top card bay on the controller and is required for operation with an IEEE 4888 GPIB interface or an RS-232C serial interface. It is also required when operating with LabVIEW drivers.

2.4.4 Expansion Card

[Included with the 11 NCS 101/IBM controller and the 11 NCS 101 expansion controller] This card fits in the bottom card bay on the controller chassis and is required for operation using the Windows control program or custom applications using Windows DLLs or the MS-DOS "C" libraries. It is also required whenever more than two Nanomover motors are to be operated from a single host computer.

2.4.5 Nanomover Motors

Nanomover motors provide linear motion according to commands issued by the user. They are self-contained units that consist of a two-phase, brushless, dc stepper motor, a micrometer screw, and a very-low-backlash coupling.

The Nanomover can provide a linear resolution of 10 nm from a micrometer screw (with a pitch of 0.5 mm per revolution) combined with a motor and driver that can provide 50,000 microsteps per revolution. It is manufactured to extremely tight tolerances using special fixtures and techniques.

The Nanomover connects to the rear of the Controller chassis with a 3-m

cable and self-locking connectors. The knob on the back of the Nanomover provides an indication of movement, and also allows for coarse manual positioning.

WARNING; Disassembly of a Nanomover will cause permanent damage, resulting in severely impaired performance. To prevent accidental disassembly of the Nanomover, the inside of the cap screws are filled with a solid compound. If these screws are tampered with in any manner, the comprehensive warranty is automatically voided.

Each Nanomover is fully tested at the factory, and a certification chart showing actual performance is included with every Nanomover. The Nanomovers are serialized so that their individual performance can be traced to the factory certification records. Removal of the serial number, located on the cable near the connector, voids the warranty.

2.5 Software

2.5.1 Nanomotion II Windows Control Program

The Nanomotion II Windows control program is a Microsoft Windows–based application that provides easy access to all of the functionality that most users will need to operate the system. The Nanomotion II control program is discussed in detail in Chapter 5, Operating the Nanomotion II System.

2.5.2 Dynamic-Link Library (DLL)

In addition to the power and flexibility of the control program described above, Nanomotion II includes a dynamic-link library (DLL) that can be used to make a custom Windows application for system control with a PC-based computer.

2.5.3 Library for MS-DOS

For users who want to construct a control application for a DOS-based IBM-type computer, a complete library of "C" functions is provided for MS C7.0 or greater. The syntax and functions found in this library are described in detail in Chapter 6, Programming Nanomotion II.

2.5.4 Command Set for IEEE 488.2/RS-232C

For users who want to control Nanomotion II using either the IEEE 488.2 (GPIB) or RS-232C (serial) interface, a set of functions is provided. These commands are contained in an EPROM on the IEEE board and can be

accessed using the ASCII-based command set provided in Chapter 6, Programming Nanomotion II or by using the supplied LabVIEW drivers.

In addition, control of Nanomotion II using RS-232C is fully backwards compatible with the original Nanomotion protocol.

2.5.5 LabVIEW Drivers (11 NCS 101/IEEE only)

Nanomotion II includes a complete set of drivers for LabVIEW for Windows, both for version 3.x and 4.x. These drivers consist of a demo program, which allows the basic setup and operation of the Nanomover, along with subsidiary virtual instrument programs (VIs) covering most operations of the Nanomover, which can be used for building unique VIs for a specific application. A listing of the VIs provided with the system can be found in Chapter 5, Operating the Nanomotion II System

2.6 Nanomotion™ II Nanopositioning System

2.6.1 Performance Parameters

While automated positioning devices have been available for many years, specification of their performance is still a source of confusion. The most useful and widely accepted set of performance parameters is that specified by the National Machine Tool Builders Association (NMTBA). The NMTBA defines three separate parameters for linear positioning devices: resolution, repeatability, and accuracy. Since these parameters are often used inappropriately, their meanings are explained below.

2.6.2 Resolution

Resolution is a measure of the capability of the system design. It is defined as the smallest move that the system is capable of making. Each movement is therefore an integer multiple of the resolution. For example, a system with a resolution of 100 nanometers is capable of moving to positions $N \times 100$ nanometers (e.g., 100, 200, or 300 nm) away from its present location. A system with a resolution of 100 nm would not be able to move to a location 150 nm from its present position. Smaller resolution provides finer gradations of movement. The resolution of the Nanomotion II system is 10 nm, which allows it to execute moves of 10, 20, 30 nm, etc., up to the limit of the travel range.

2.6.3 Repeatability

Repeatability is a measure of one type of system error — the repositioning error. It is defined as the error within which a given position can be reproduced. The smaller the value for repeatability, the better the system.

Specifically, repeatability is the difference in absolute position that occurs when the system is moved to the same point at several different times. Unidirectional and bidirectional repeatability can be separately defined. Unidirectional refers to a situation where the target position is always approached from the same side. Bidirectional repeatability is typically much worse than unidirectional repeatability because of hysteresis and backlash effects. Nanomotion II has a guaranteed bidirectional repeatability of 100nm.

2.6.4 Accuracy

Accuracy is a measure of another type of system error — the absolute positioning error. It is defined as the absolute deviation between the target position and the actual position. Thus, accuracy error is a measure of the ability of the system to move exactly to an absolute position. Accuracy is generally not as important as repeatability, since consistent, repeatable accuracy errors can be evaluated and compensated. Nanomotion II has an accuracy of $\pm 1\ \mu$m, most of which results from minute variations in the micrometer thread pitch and unavoidable inaccuracies in the stepper motor. If you need extremely high accuracy, see the calibration graph in the appendix for guidance.

2.7 Nanomotion™ II Concepts

2.7.1 Travel Line

The travel line is the range of motion over which the Nanomover can move. Each Nanomover can be positioned anywhere along the 25 mm long travel line, at a resolution of 10nm with a bidirectional repeatability of 100 nm.

2.7.2 Unit Independence

The travel line is always considered in terms of 10-nm increments. The Nanomotion II software converts this 10-nm unit into most common units so that the user can program a Nanomover in inches, mils, microinches, centimeters, millimeters, microns, or nanometers. The Nanomotion II Control Windows Program also allows the user to define custom units for individual applications.

2.7.3 Direction of Motion

The Nanomover is capable of moving in two directions, as follows: a positive (+) motion indicates an extension of the micrometer shaft; a negative (−) motion indicates a retraction of the shaft. In addition, for the purposes of this manual, (+) refers to a movement to the right on the travel line, and (−) refers to a movement to the left on the travel line.

2.7.4 Velocity Profile

The velocity profile describes the relationship between speed and position as the Nanomover is making a move. The profile is broken up into four parts — base velocity, acceleration and deceleration, slew velocity, and position-lock time. In general, a Nanomover starts at a base velocity, accelerates to the slew velocity using one or more acceleration ramps, decelerates back to the base velocity, and then stops.

2.7.5 Acceleration and Deceleration Ramps

The Nanomover can operate at speeds up to 2.5 mm/sec (2,000 full motor steps per second), but it cannot start or stop while moving at this high rate of speed. Instead, it must accelerate from an initial velocity until it reaches the specified slew velocity, and then decelerate to a final velocity before stopping.

The Nanomotion II system divides the acceleration and deceleration ramps into four individually controllable segments (the deceleration curve is the reverse of the acceleration curve). The use of four segments allows a reasonable approximation of the ideal, exponentially-shaped, acceleration profile.

Each of the four segments is defined in terms of velocity and acceleration. To move, the Nanomover accelerates at its defined rate until it reaches the specified velocity. When this velocity is reached, the motor accelerates with the value in the next segment, continuing this process until the movement is complete.

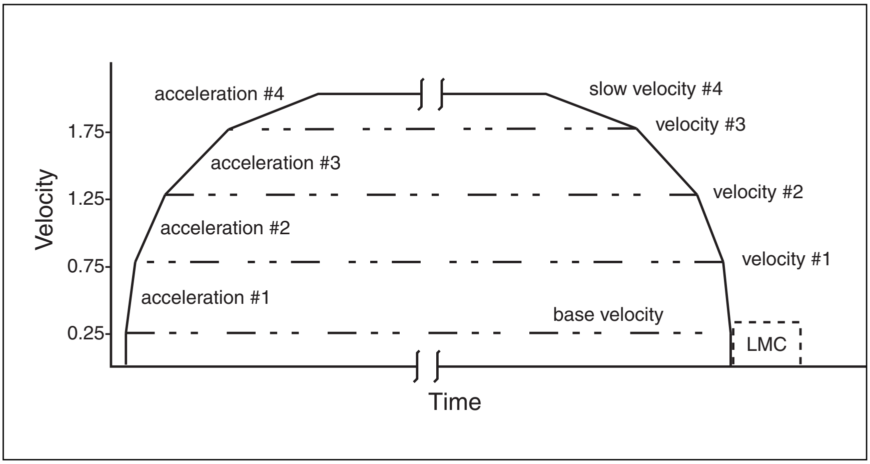


Figure 2.3 Velocity/Acceleration Ramping

In the example shown in the figure, the Nanomover starts at a base velocity of 0.25 mm/sec. The acceleration for the first ramp segment occurs at 400 mm/sec^2 until a velocity of 0.75 mm/sec is reached. The next ramp uses an acceleration of 200 mm/sec^2 to a velocity of 1.25 mm/sec, the third ramp uses an acceleration of 125 mm/sec^2 to 1.75 mm/sec, and the fourth ramp uses 100 mm/sec^2, to a final velocity of 2.00 mm/sec. Although the above example uses millimeters as a generic unit, any common measurement unit can be used.

Not all ramp segments must be executed. The Nanomotion II system will stop acceleration whenever a velocity value is lower than the previous ramp velocity or if it encounters an acceleration value of zero. In a 'trapezoidal' motion profile, only one acceleration ramp is used to reach slew velocity.

It is interesting to note that, for many applications, there is a negligible difference in performance between using all four acceleration ramps and using only one ramp. In fact, the default acceleration values use only one ramp. However, there are applications involving heavy loads, or loads with high inertia, where performance will vary significantly with changes in the acceleration curves. Other applications may need to have movements occur very quickly or very smoothly. Redefining the acceleration curves can improve system performance for all of these situations.

Units (per second)	Minimum Velocity	Maximum Velocity
Nanometers (nm)	5,000	2,500,000
Micrometers (μm)	5	2,500
Millimeters (mm)	0.00	2.50
Centimeters (cm)	0.05	.250
Microinches (μin.)	197	984,000
Mils (1/1000 in)	1.97	984
Inches (in.)	0.00197	0.984

Table 2.1 Nanomotion II Velocity Range

For most loads, the maximum Nanomover acceleration should be less than 1250 mm/sec^2. The accelerations specified in the acceleration ramp must be within the following ranges.

Units	Maximum Acceleration
Nanometers (nm/sec^2)	1,250,000,000
Micrometers ($\mu m/sec^2$)	1,250,000
Millimeters (mm/sec^2)	1,250
Centimeters (cm/sec^2)	125
Microinches ($\mu in./sec^2$)	49,210,000
Mils ($mils/sec^2$)	49,210
Inches ($in./sec^2$)	49.21

Table 2.2 Nanomotion II Maximum Acceleration

2.7.6 Base Velocity

As mentioned above, the motor starts and stops from a preselected speed, called the base velocity. The base velocity can range between 0.005 to 0.1875 mm/sec. For most applications, this parameter can be kept as high as is appropriate for the mechanical system being moved. Note that the lower of the base velocity or the first ramp velocity will be used as the starting velocity.

2.7.7 Dual Current Levels

One of the many novel technological features of Nanomotion II involves the energy provided to the motors. Unlike most stepper motor systems, Nanomotion II can supply two different levels of current to the motors. Two current levels are ideal because the torque requirements to hold the motor in position are considerably less than the torque requirements to move the motor. Extra energy applied during periods of no movement must be dissipated as heat, resulting in thermally induced positioning errors. Because Nanomotion II can provide two current levels, this error can be minimized. The default stepping and holding currents are 900 mA and 400 mA, respectively.

2.7.8 Position Lock-Time

The position lock-time is the length of time that the motor current is held high, after movement is completed, before it is reduced to the holding current level. A non-zero position lock-time is necessary to control the inertia contained within a decelerating system. In other words, for the motor to stop quickly and still maintain position, it must have as much current available as possible. Therefore, the current is maintained at the stepping

current level until the motor can "lock" itself into position. The current is then reduced to the holding level.

Position lock-time can range between 0 and 255 msec. Generally, a value of around 10 msec is adequate for the Nanomover. Longer times should be used for heavy loads, or loads with high inertia.

2.7.9 Lost Motion Compensation (LMC)

In any mechanical system, there are small clearances, or tolerances, needed for parts to move. These small clearances combine and manifest themselves as a dead space when a small move is attempted, or when the direction of motion is reversed. Another feature of mechanical systems is stiction, which is the static friction that must be overcome to start a movement.

Lost motion is defined as requested movement that does not result in mechanical motion due to dead space or stiction effects. When the dead space has been removed from the system, and enough force has been developed to overcome stiction, the motion will occur. Lost motion is a primary cause of the error described as bidirectional repeatability. Lost motion compensation (LMC), is the novel method used by Nanomotion II to compensate for this error.

If LMC is enabled by the user, the bidirectional repeatability can be equal to or even exceed the unidirectional repeatability. This is achieved by always approaching the target position from the same side with a final move that is always the same in every aspect. Approaching the target from the same side is a technique that has been used for quite some time. However, by requiring that the final move always be the same length, all lost motion is consistently removed, and new levels of bi-directional repeatability can be achieved.

The magnitude and direction of LMC are selected by the user. Values from $-10\ \mu m$ to $+10\ \mu m$ can be selected, where negative values refer to a final approach in the negative direction, and positive values refer to a final approach in the positive direction. If the move is in the direction opposite that for the LMC, then the final LMC move will be preceded by a target "overshoot." Similarly, if the overall move is in the same direction as the LMC, then the final LMC move will be preceded by a target "undershoot."

Having a choice of direction of LMC can be very useful, particularly where there is a definite constraint against shaft overshoot, as in the mechanical probing of surfaces (e.g., semiconductor wafers). Only one direction of LMC is acceptable in these cases in order to avoid damaging the surface.

If LMC is enabled, undershoot at the absolute left stop and overshoot at the absolute right stop may occur during the LMC move. The maximum

magnitude of this under/overshoot is the LMC amount. Usually this will not present any problem, although it may need to be taken into account when setting absolute stops.

NOTE: Each unit of LMC corresponds to a final move of 10 microns. Values between 0 and 5 are recommended for most applications.

2.7.10 Stopping Movement

When a Nanomover executes a requested move, it slows down using deceleration ramps and concludes movement with a Lost Motion Compensation move, if LMC has been enabled. However, there are other circumstances under which a Nanomover may stop moving, and the details of these need to be considered.

If a Nanomover reaches an Absolute Stop, then the Nanomover will come to a halt using the specified deceleration ramps and LMC. Position memory is maintained and no “lost motion” (i.e., backlash) is accumulated. Depending on the sign of the LMC, a Nanomover may momentarily move beyond an Absolute Stop. This is the only circumstance under which an Absolute Stop position may be exceeded.

If a Nanomover reaches a limit switch the Nanomover will stop according to the specified deceleration ramps. While no LMC is used, position memory is retained. Some "lost motion" may accumulate, but this will disappear on the next move. If a Nanomover is stopped by a software STOP command, it will stop as if it had reached a limit switch.

2.7.11 Park and Unpark

The PARK and UNPARK features provide a mechanism to maintain the Nanomover position even when power is turned off. The PARK command instructs a Nanomover to move to its closest stable position, where the position will be maintained even when the power is turned off. The movement necessary to reach that position is recorded in a file, and then the power is smoothly removed from the Nanomover in order to prevent any jumping or slipping. The system can then be turned off. When the system is powered, the command UNPARK will read the position file and restore the Nanomover to its initial position.

If lost motion compensation is enabled, UNPARK will restore the motor using the specified LMC sequence.

NOTE: Any movement of the Nanomover shaft while power is off will cause the system to be incorrectly restored and synchronization to be lost.

2.8 Location Concepts

2.8.1 Zero Position

The Zero position is the location on the travel line from which all other positions are normally referenced. It defines the beginning of the travel range. It is also possible to reference all positions (except for the absolute stops) with respect to a variable Home position.

2.8.2 Absolute, Relative, and Home Positions

Absolute and relative positions are similar, but differ in the starting location on which they are based. While absolute positions are located with respect to the zero position, relative positions are located with respect to the home position. Because the home position can be located anywhere with respect to the zero position, relative positions provide a convenient means of locating a unique movement index.

2.8.3 Stops

Stops are used to confine the range of travel to a range shorter than the maximum possible. This feature is often used in applications that do not require the full travel range. The limits are set by the absolute left stop and the absolute right stop, which are expressed relative to the zero position. It is recommended, but not required, that the zero and absolute left stop be set in the same position, so that all accessible positions will have positive values. The system will never allow a Nanomover to position itself beyond the values set by the absolute left and absolute right stops, except for LMC movement as described in subparagraph 2.7.9, Lost Motion Compensation.

2.8.4 Defining the Axes

Each Nanomover is defined and operated in terms of an axis. Up to 16 axes can be supported by the software. Therefore, because each Nanomover can be controlled by more than one (user-defined) axis, multiple types of motion can be assigned to the same Nanomover.

2.8.5 Limit Switches

For a particular setup, Nanomovers can be protected against movement beyond a safe range by two sets of software limits. However, some users may wish to install external limit switches to provide additional safeguard input to the system to protect sensitive components or instruments. Nanomover systems are configured for limit switches of the normally-open switch contact or opto-interrupt type. Whenever a closed limit switch is detected by a Nanomover, the Nanomover will not be driven any farther in that direction (see Chapter 3, Specifications).

2.8.6 Position Synchronization

The Nanomovers are open loop devices. Extensive tests show that Nanomotion II systems exhibit perfect current pulse to motor step repeatability. The microcontroller for each Nanomover always knows the position of the Nanomover from the values stored in the position counters. When a system is initialized, it is necessary to synchronize these counters with the Nanomover micrometer shaft — that is, to ensure that the zero and 25 mm in the counters represent zero and 25 mm of shaft extension.

There are three ways in which the system can be synchronized;

1. On power-up, the microcontrollers assume that all Nanomovers are located at the zero location. The Nanomovers are correctly synchronized if they have each been manually set with 1.5 mm of shaft extension and the manual location knob has been turned to point directly away from the cable.
2. Before the user exits the Nanomotion II system, the PARK command can be used to park any or all of the motors. When the system is powered-up and the software is reinitialized, the user has the option to use the UNPARK command to retrieve the previous position(s) and maintain the synchronization (± 100 nm).
3. The synchronization of any of the Nanomovers can be reset at any time using the WRITE POSITION command. This command can be used to reset the position counter to the current physical displacement of the Nanomover micrometer shaft.

Note: The synchronization position does not need to use the recommended zero position with 1.5 mm of shaft extension. Other locations can be used. However, the absolute stops must then be carefully applied to ensure that the Nanomover does not exceed its normal range of travel.

2.8.7 Default Movement Parameters

The movement characteristics of Nanomotion II are ordinarily determined by user-specified values for the parameters available on the motor parameters page. If one or more of these parameters is not defined by the user, then the parameter will be given a default value. The default values have been calculated for average conditions and should be adequate for most applications. These parameters and their default values are listed on the following page.

Parameter	**Default Value**
Base Velocity	.0625 mm/sec
Units to Move	+1.0 mm
Acceleration	500 mm/sec^2
Velocity	2.0 mm/sec
Absolute Right Stop	25.4 mm
Absolute Left Stop	0.0 mm
Lost Motion Compensation	Enabled at +0.05 mm

Table 2.3 Nanomotion II Default Parameters

(This page intensionally left blank.)

Chapter 3

Specifications

3.1 Nanomover Motor Specifications

Characteristic	11 NCM 001	11 NCM 005	11 NCM 007
Tip	flat	flat	spherical
Resolution		10 nm	
Bidirectional Repeatability		±100 nm	
Absolute Accuracy		±1 μm	
Calibration Standard		H.P. 5528A laser interferometer to NBS reference	
Maximum Speed		2.5 mm/sec	
Travel Range		25 mm	
Maximum Load	10 kg mass	20 kg mass	10 kg mass
Maximum Acceleration		1250 mm/sec^2	
Standard Cable Length		3 m	
Maximum Cable Length		35 m	
Stepping Current	0.9 A	1.2 A	0.9 A
Holding Current	0.45 A	0.6 A	0.45 A
Steps per Revolution		400	
Voltage Required		4 Vdc	
Holding torque		78 N-cm	
Physical Dimensions		44.5 mm × 44.5 mm × 158 mm	
Mounting Barrel Diameter		10 mm + 0.0/ −0.009 mm	
Weight	370 g	500 g	370 g
Operating Temperature		+18°C to +40°C	
Storage Temperature		−40°C to +70°C	

3.2 Nanomotion™ II Controller Chassis Specifications

Amplifiers per Chassis	2
Maximum Amplifier Voltage	±21.5 V
Maximum Current per Amplifier	2 A
Bus Compatibility	IEEE and RS 232
Minimum Speed	5 mm/s (4 full steps/sec)
Maximum Speed	6.25 mm/sec (500 full steps/sec)
Maximum Acceleration	1,000,000 full steps/sec^2
Acceleration/Deceleration Ramp Segments	4
Number of axes supported	16
Safety Ratings	Designed per U/L, CSA, TUV, and VDE specifications
Input Voltages	90-130 Vac, or 200-240 Vac, at 47-63 Hz
5-V Power Supply	8 A maximum
±12-V Power Supply	400 mA maximum
Physical Dimensions	215 mm × 326 mm × 100 mm
Limit Switch Types	Opto-interrupter or contact-type switches (normally open)
Weight	4.4 kg
Operating Temperature	+10°C to +40°C
Storage Temperature	−40°C to +70°C

3.3 Controls, Indicators, and Connectors

3.3.1 Front Panel Controls and Indicators

The front panel of the Nanomotion II controller is shown in Figure 3.1. The panel contains one control (an ON/OFF switch) located in the lower right-hand corner of the unit, and a power-on indicator light (a green LED) in the upper right-hand corner.

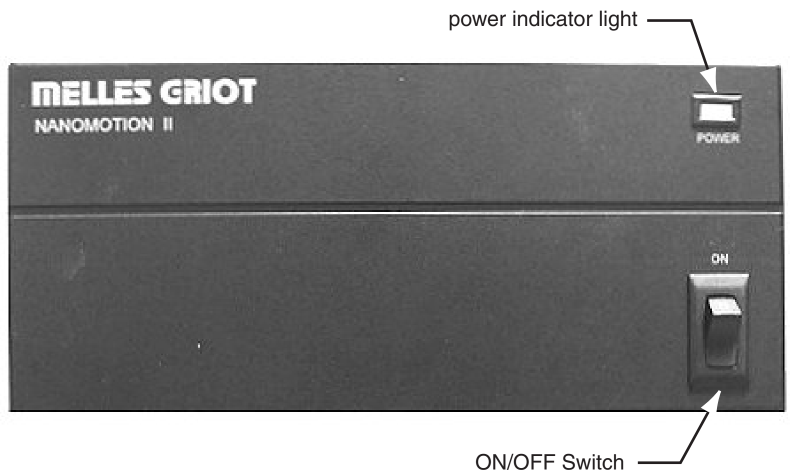


Figure 3.1 Front Panel of Nanomotion II

3.3.2 Back Panel Controls and Connectors

The back panel of the Nanomotion II controller, shown in Figure 3.2 contains three horizontal card bays. The top bay contains the IEEE/RS-232 interface board, the middle bay contains the motor control board, and the bottom bay contains the expansion board used to daisy-chain controllers. There are three possible configurations for the unit.

- Standard configuration for IEEE 488.2/RS-232C controller (11 NCS 101/IEEE): In this configuration, the top bay and center bay are filled. The bottom bay contains a blank panel. This configuration is limited to the use of two motors.
- Standard IEEE 488.2/RS-232C configuration shown in Figure 3.2 with expansion board (11 NCS 101/IEEE with 11 NIB 001): In this configuration, all bays are filled. The unit can now be daisy-chained to other controllers for operation of up to 16 motors. With appropriate software and a PC-Link card, it can also be used with the Windows Control Program, DLLs, and MS-DOS "C" libraries.
- Standard Configuration for IBM-compatible computers. (11 NCS 101/IBM). In this configuration, the top bay contains a blank panel and the bottom bay contains the IBM ISA interface board. A PC-Link card, the Windows Control Program, DLL library and "C" libraries are included with this configuration.

- Expansion controller (11 NCS 101): This configuration is daisy-chained to the standard controller (11 NCS 101/IEEE) to operate additional units. In this configuration, the top bay is empty, and the bottom bay contains the 11 NIB 001 expansion card. It is essentially the same as the standard IBM-compatible configuration described above, but it does not include the PC-Link card or software.

All chassis have a motor control board located in the middle slot. This is where limit switches for the motors are connected (use of limit switches is optional) and where bus addressing is done for multiple axis systems using one IEEE address.

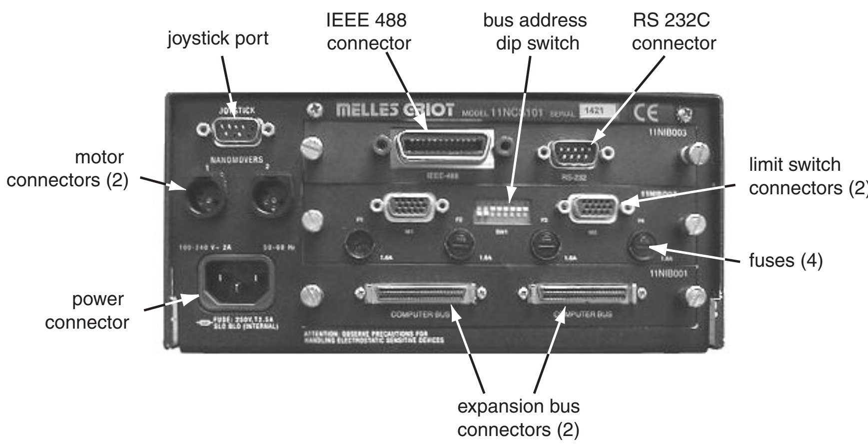


Figure 3.2 Back Panel of Nanomotion II

Limit switch connectors (middle bay)

Limit switches are not necessary with Nanomotion II, which is protected by software travel limits. However, software travel limits are user-set and mistakes can be made by inexperienced operators. While the Nanomovers themselves cannot be damaged if driven beyond the travel range, limit switches are recommended to protect the delicate and expensive equipment devices often used with Nanomotion II.

Nanomotion II can support up to two limit switches (left and right limits) for each Nanomover. The two most common types of limit switch are contact and opto-interrupt. Both are supported in the normally-open configuration (i.e., the switch is closed to complete the circuit and prevent further travel).

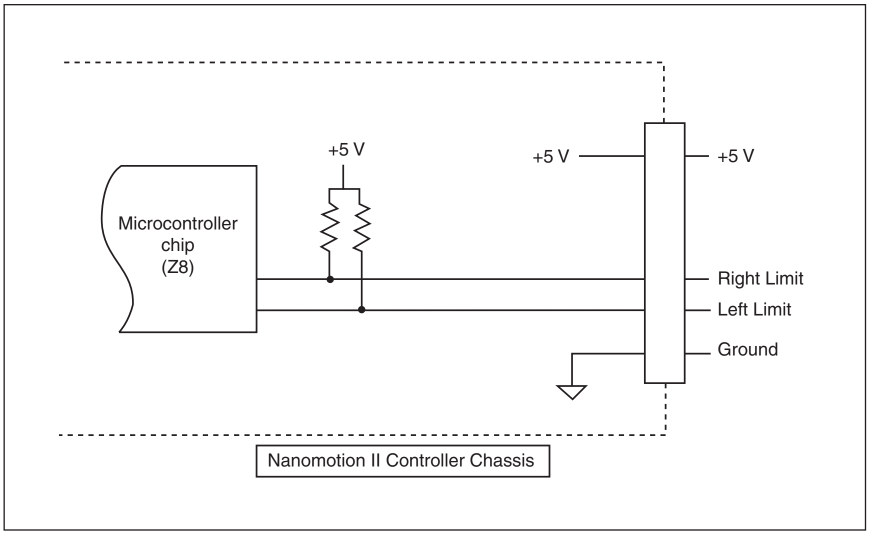


Figure 3.3 The limit switch interface (one channel shown).

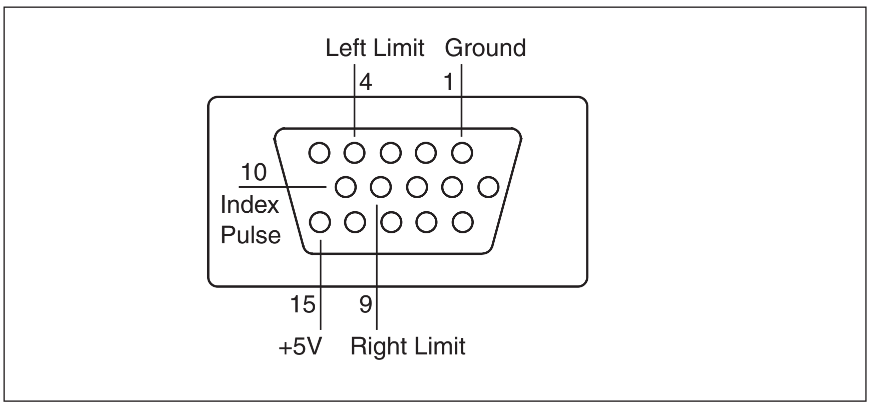


Figure 3.4 Pin assignments for the limit switch interface.

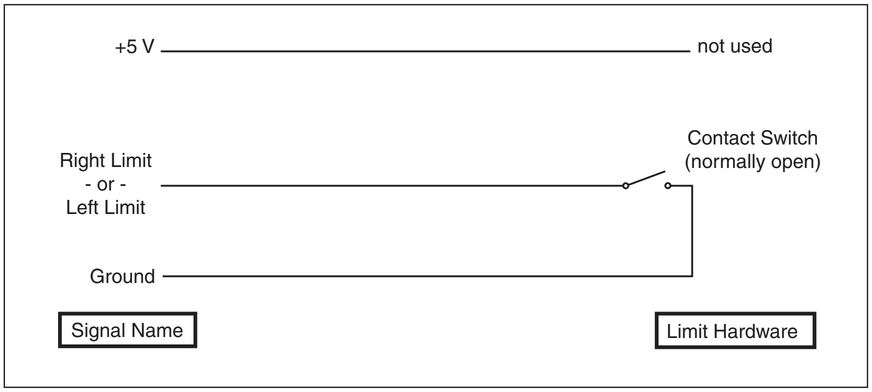


Figure 3.5 Connections for a contact switch.

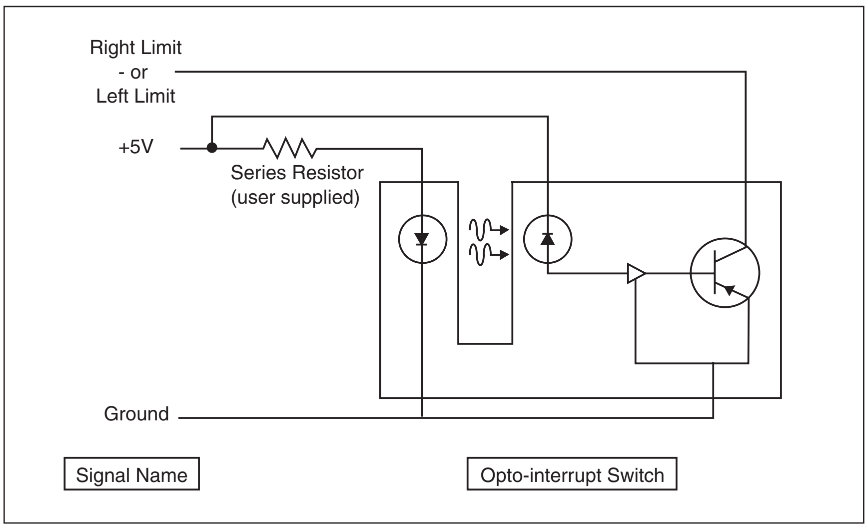


Figure 3.6 Connections for an opto-interrupter switch.

The limit switch connectors are located on the back of the Nanomotion II controller chassis. When using limit switches, it is very important that the switch cables not be crossed, so that they correspond to the correct Nanomover.

Bus address dip switch (middle bay)

The settings of the 8-pin DIP switch define an address that allows the Nanomotion II system to identify each set of axes that are daisy-chained together. The settings of this switch should not be changed from the factory default for IEEE systems unless required for multiaxis operation. See Chapter 4, Installation and Setup for more details.

IEEE connector (top bay)

The IEEE GPIB connector is located on the left-hand side of the card in the upper bay. The mating GPIB cable is not supplied with the system. GPIB cables are available from computer suppliers. (This connector is not present on the 11 NCS 101 expansion chassis.)

RS-232-C connector (top bay)

The 9-pin RS-232C serial connector is located on the right-hand side of the card in the top bay. The mating serial cable is not supplied with the system. These are available from computer suppliers. (This connector is not present on the 11 NCS 101 expansion chassis.)

Expansion bus connectors (bottom bay)

The two SCSI-2 expansion cables are located on the bottom bay. A cable for connecting two chassis together should be included with any chassis system with these connectors. Cables can be attached to either connector on this board, in any order. The cable is a standard SCSI-2 type available from computer suppliers. Detailed instructions on connecting the expansion chassis with the IEEE system is found in Chapter 4, Installation and Setup. (These connectors are not present on the 11 NCS 101/IEEE unless the 11 NIB 001 expansion card has been ordered.)

Joystick (main chassis)

The 9-pin connector in the upper left-hand corner of the back panel is used to plug in the 11 NCA 101 joystick. On IBM systems, the software can be modified to use the joystick instead of hot keys on the PC keyboard. On IEEE systems, the joystick can operate the system without being connected to a host computer.

Motor connectors (main chassis)

Nanomover motors are plugged into the two connectors located directly below the joystick connector. The connector on the left is motor #1 on the first chassis, and the connector on the right is motor #2. On succeeding chassis in a daisy chain, the left connector is always the lowest numbered motor (3,5,7, etc.) and the right connector is the higher numbered motor (4,6,8, etc.)

Power connector (main chassis)

This is a standard 3-prong plug. A power cable for U. S. usage is included with each system. The chassis uses a universal input power supply that will accept 90–240 V at 47–63 Hz.

Fuses (main chassis)

The fuse cartridge is located in the power cord input connector inside the chassis. This fuse has a slo-blow rating of 250 V, T2.5AH, and is a universally available size (5 mm $\times$ 20 mm).

Fuses (motor control board)

There are 4 fuses located on the back panel of the motor control board. These fuses have a fast-blow rating at 250 V, 1.6 A and are easily accessible from their twist-lock holders.

Chapter 4

Installation and Setup

4.1 Parts List

Name	MG Part Number	Description
Nanomotion II Controller Chassis for GPIB or Serial Interface	11 NCS 101/IEEE	Contains various electronics and boards to communicate via IEEE and RS-232C interfaces and to control up to two Nanomovers
Nanomotion II Controller Chassis for IBM-Compatible Computers	11 NCS 101/IBM	Contains various electronics and boards to communicate via a PC-Link card with Windows and MS-DOS-based programs. Includes PC-Link card, DLL libraries, Windows, and DOS software.
Nanomotion II Expansion Chassis	11 NCS 101	Contains various electronics and boards to communicate with other controller chassis and to control up to two additional Nanomovers. Includes expansion cable for daisy chaining to other 11 NCS 101 chassis.
IBM Interface Board	11 NIB 001	Installed in controller chassis, interfaces chassis to a computer via a PC-Link card. Also used for daisy-chaining controllers when more than two Nanomover motors are used. Standard with 11 NCS 101/IBM and 11 NCS 101. Optional with 11 NCS 101/IEEE.
IEEE RS-232C Interface Board	11 NIB 003	Installed in controller chassis, interfaces chassis to a computer via the IEEE 488.2 or RS-232C port. Included in 11 NCS 101/IEEE.

Name	MG Part Number	Description
PC-Link Card	11 NIB 005	For installation into IBM-PC (AT or later) to interface to controller. Includes software. Standard with 11 NCS 101/IBM. Optional with 11 NCS 101/IEEE.
Motor Control Board	11 NIB 007	Installed in all controller chassis, contains Nanomover control electronics and amplifiers.
Power Cord	Included with 11 NCS 101 11 NCS 101/IBM 11 NCS 101/IEEE	ac power cord for Nanomotion II Controller Chassis.
Expansion Cable	Included with 11 NCS 101 and 11 NIB 005	Connects Nanomotion II controller chassis to expansion chassis.
Nanomover (standard)	11 NCM 001	Motorized micrometer with standard specifications.
Nanomover (high torque)	11 NCM`005	Motorized micrometer with high torque specifications.
Nanomover (spherical tip)	11 NCM 007	Motorized micrometer with a spherical tip for use with stages, etc. without ball bearing connections.
Nanomover Extension Cable	11 NCA 003	Three-meter cable that extends the distance between Nanomover and controller chassis.
Joystick	11 NCA 101	Allows operation of motors by joystick control. Does not require computer connection.
System Manual	22 MAN 101 Rev C	This manual

4.2 General Precautions

Each Nanomotion II system includes precision mechanical and electronic components that will have a useful life of many years. Certain precautions must be observed to ensure that these components continue to operate according to the specifications.

4.2.1 Power Supply

The Nanomotion II Controller Chassis must be connected to a grounded power line. Do not use an adapter plug designed to defeat the third, grounded prong. While the Controller Chassis contains electronics to protect it from power-line surges, it is recommended that the chassis not be connected on the same circuit breaker as any electrically-noisy equipment.

WARNING: Always unplug the power cord when servicing fuses in order to prevent accidental electric shock.

The chassis uses a universal input power supply that will accept 90-240 volts at 50-60 Hz.

4.2.2 Electrical Safeguards

The Nanomotion II cards must be protected from static electricity when they are not mounted in a grounded chassis. You must be grounded when handling the cards. Once they are mounted in a chassis, the cards and the entire system are afforded ESD protection typical for steel enclosed electronic cabinets.

WARNING: Nanomotion II cards should never be inserted or removed from a Controller Chassis or PC chassis unless all power to that chassis is switched off.

4.2.3 Location

The Nanomotion II Controller Chassis should be mounted on a horizontal surface, away from any sources of dust and vibration. The chassis should also be protected from direct sunlight, excessive heat, and moisture. The air slots for the cooling fan, located on the side of the chassis, must remain unblocked or the chassis will overheat and the thermal interlocks will cause the unit to shut down. The Nanomover(s) should also be isolated from excessive dust, and must be vibration-free to operate at a high resolution.

4.2.4 Handling

Nanomotion II components, especially the Nanomovers, must not be subjected to physical abuse. If a Nanomover is dropped, it will almost certainly be permanently damaged.

> **WARNING: Servicing of nanomovers must be performed by factory personnel only. Opening or tampering with the nanomover end plate will void the product warranty.**

4.2.5 Vibration Isolation

The use of a vibration isolation workstation is strongly recommended to fully exploit the capabilities of Nanomotion II.

4.2.6 Lubrication

The Nanomover uses special lubrication that must be kept free of particulates to ensure that the high-precision micrometer screw operates without errors. Any particles of lint or dust that become trapped in the screw mechanism will cause errors or permanent damage.

During the first 100 hours of use, some lubrication may collect on the micrometer shaft. This excess lubrication should be carefully removed with a lint-free material such as lens tissue.

4.2.7 Temperature

Nanomovers must be kept at a stable temperature when being operated. Differences in temperature can cause changes in the length of metal parts in the positioning system, affecting system performance. While Nanomover is designed so that thermal expansion of only the relatively short micrometer shaft can affect the system precision, many of the translation stages and other components often used with the Nanomotion II system are constructed of metal and are therefore subject to thermal expansion of an indeterminate amount. The following table summarizes the thermal expansion of metals commonly used in optical mounting accessories.

Metal	Expansion, per °C, of 25 mm of Metal
Stainless Steel	452 nm
Aluminum	578 nm
Brass	470 nm

4.3 Installing the IBM System

4.3.1 Hardware Installation

1. Read the general precautions covered in section 4.2.
2. If you are installing more than one chassis, it is recommended that you get each chassis to work as a single system prior to setting up as a multiple-axis system.
3. Hook up power, hook Nanomover motor cable to the motor connections on the back. If you have a joystick, this can be plugged in and used to check movement of the motors and/or stages.
4. Install the PC-Link board in the bus slot of the host IBM-compatible computer. Consult your PC manual for special instructions regarding inserting boards, and remember to remove the cover plate over the appropriate slot so that the cable connector is accessible.
5. Connect the PC-Link card to the controller chassis using the cable provided.This cable is terminated on both ends with a 50-pin "D" type connector. The cable is physically keyed by the "D" connector so that it cannot be inserted incorrectly. The cable can be attached to either of the two SCSI-2 connectors on the rear panel of the controller chassis.
6. If additional axes need to be used, refer to section 4.4.7 for detailed instructions.

4.3.2 PC Software Installation

Windows Systems

If you use the Windows operating system, insert the disk labeled Nanomotion II Software into the floppy disk drive. Start Windows, then, under the File menu, select Run and select your floppy disk drive. Then select "Setup". Choose a location to install the software and verify that a program group entitled "Nanomotion II" is created. This completes the software installation.

Test the software by double-clicking on the Nanomotion II icon. The Nanomotion II welcome window should appear.

DOS Systems

If you use the DOS operating system, insert the disk labeled Nanomotion II Software into the floppy disk drive. Type a:setup and respond to the prompts to select an installation path for the software.You will be prompted throughout the installation process.

Installed Files

The following files will be installed by either of the installation routines described above.

File	Function
nanomotn.EXE	Executable Nanomotion II Control Program
nanomotn.HLP	Nanomotion II Help File
nonomotn.INI	Initialization File
demo.AXS	Example Axis File
nanodll.DLL, nanodll.lib	Nanomotion II DLL for Windows Applications
nanolib.lib	Nanomotion II Library File
Nanolib.H	C Library Include Files
	LabVIEW Driver Files

4.3.3 Connecting a Joystick

On IBM systems, the software can be modified to use the joystick instead of the hot keys on the PC keyboard. Note that a host computer must be used with the joystick on an IBM system.

4.3.4 Setting the PC Addresses

When multiple chassis are daisy chained together, each board must be assigned a unique address so that information can be exchanged without becoming lost or misdirected. The addresses are assigned using a series of 8 DIP switches, together referred to as SW1, located on the back of the Controller Chassis.

Use the table below to determine the correct SW1 switch settings for your Nanomotion II board(s). Each board can control two Nanomovers that are referenced in the software by their motor numbers. It is recommended that the IEEE chassis remain at address 300, and that all additional units be in contiguous addresses above it.

NANOMOTION II BOARD ADDRESSES

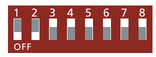
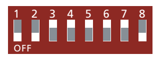
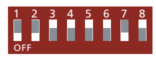
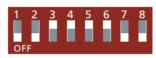
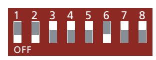
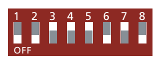
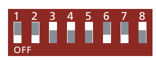
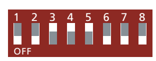
Motor Numbers	Address	Switch Setting
1 and 2	300	
3 and 4	304	
5 and 6	308	
7 and 8	30C	
9 and 10	310	
11 and 12	314	
13 and 14	318	
15 and 16	31C	

Figure 4.1 Motor Board Addresses

4.4 Installing the IEEE 488.2/RS-232C System

4.4.1 IEEE 488.2 (GPIB) System

1. Read the general precautions covered in section 4.2.
2. If you are installing more than one chassis, it is recommended that you get each chassis to work as a single system prior to setting up as a multiple-axis system.
3. The default GPIB address is 5. If this must be changed, refer to section 4.4.3 for details on changing the GPIB address.
4. Hook up power, hook Nanomover motor cable to the motor connections on the back. If you have a joystick, this can be plugged in and used to check movement of the motors and/or stages.
5. Install a GPIB cable (not included).

6. Install the LabVIEW software (see section 5.2) or use a dialog program such as National Instrument's IBIC to check the setup of the system. Use the MR command (see Chapter 6, Software Commands) to move the motors.
7. If additional axes need to be used, refer to section 4.4.7 for detailed instructions.

4.4.2 RS-232C (Serial) System

1. Read the general precautions covered in section 4.2.
2. If you are installing more than one chassis, it is recommended that you get each chassis to work as a single system prior to setting up as a multiple-axis system.
3. The default serial interface is shown below. If this must be changed, refer to section 4.4.3 for details on changing it.

Baud Rate	19200
Handshaking	Disabled
Stop Bits	1
Parity	None
Data Bits	8

4. Hook up power, hook Nanomover motor cable to the motor connections on the back. If you have a joystick, this can be plugged in and used to check movement of the motors and/or stages.
5. Install a 9-pin serial cable (not included).
6. No software is loaded for the serial system. Use something like Windows hyper-terminal software to check the setup of the system. Use the MR command (see Chapter 6, Software Commands) to move the motors.
7. If you need to hook up a multiple-axis system, contact your local Melles Griot technical support person for detailed instructions.

4.4.3 Configuring the IEEE 488.2/RS-232C Interface Board

There are three dipswitches on this board that may need to be changed for your configuration (see Figure 4.2 for location of these switches). To access these dipswitches, loosen the two screws that hold the IEEE board in place, and pull the board out using even pressure on both sides. It is a good idea to record any changes made to these dipswitches for future reference, and to avoid handling the board any more than is necessary. After making

changes to the dipswitches, insert the board into the slot, ensuring that it is seated correctly.The system can then be connected via the IEEE 488 or Serial buses.

> **WARNING: Be sure to wear a grounding strap or take appropriate precautions to avoid damaging the Nanomotion II boards with static electricity. When removing or replacing boards in the Controller Chassis be sure that the power is turned off. Removing or replacing boards with power on will cause them to be damaged.**

Record your settings here (if different from factory settings)

	Switch 1 (Serial)								**Switch 2 (GPIB)**								**Switch 3 (Joystick)**							
Switch Settings	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
On																								
Off																								

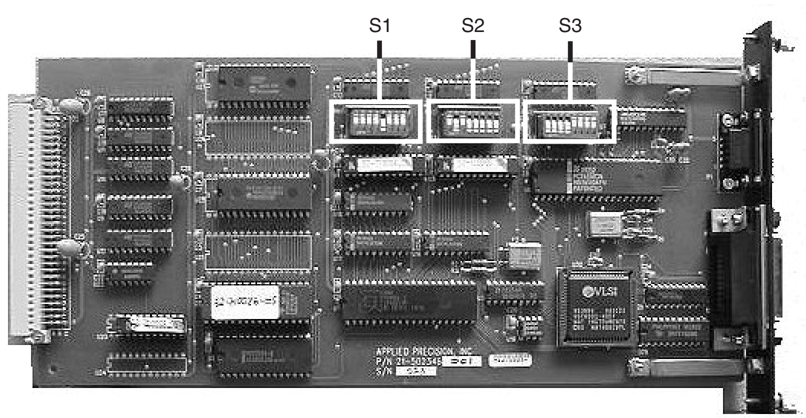


Figure 4.2 IEEE board

Switch S1 - Serial Configuration

Factory Settings:

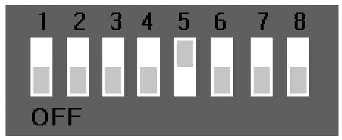


Figure 4.3 Serial Configuration Switch

Pin 1	Pin 2	Pin 3	Baud Rate
Off	**Off**	**Off**	19200 (Factory setting)
On	Off	Off	9600
Off	On	Off	4800
On	On	Off	2400
Off	Off	On	1200
On	Off	On	600
Off	On	On	300
On	On	On	110

Pin 4	Handshaking
Off	Disabled (Factory setting)
On	Enabled

Pin 5	Stop Bits
Off	Two
On	One (Factory setting)

Pin 6	Pin 7	Parity
Off	**Off**	None (Factory setting)
On	Off	Odd
On	On	Even

Pin 8	Data Bits
Off	Eight (Factory Setting)
On	Seven

Switch S2 - GPIB Address

Factory Settings:

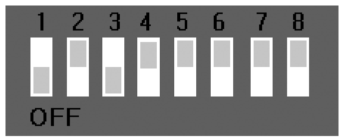


Figure 4.4 GPIB Address Switch

This switch uses binary code to determine the GPIB Address for the device. Bit 8 is the most significant and bit 1 is the least significant. Settings from 1 through 31 (decimal) are recognized. The factory setting is 5 (shown above).

SW 1	SW 2	SW 3	SW 4	SW 5	SW 6	SW 7	SW 8	GPIB Address
off	on	on	on	on	on	on	on	1
on	off	on	on	on	on	on	on	2
off	off	on	on	on	on	on	on	3
on	on	off	on	on	on	on	on	4
off	**on**	**off**	**on**	**on**	**on**	**on**	**on**	**5**
on	off	off	on	on	on	on	on	6
off	off	off	on	on	on	on	on	7
on	on	on	off	on	on	on	on	8
off	on	on	off	on	on	on	on	9
on	on	on	off	on	on	on	on	10

Switch S3 – Joystick Switches

The joystick operation allows movement via the x-y control pad and extra movement when the control pad is used while the "B" button is pressed. The movement achieved by operation of the control pad is configurable, as is the response associated with pressing the "B" button. Bits 1-2 cover the movement due to the x-y control pad. Bits 3-8 cover the parameters of the move and the "B" button response. See section 4.4.6 about connecting the joystick.

Factory Settings:

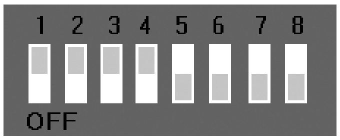


Figure 4.5 Joystick Switches

The factory setting sets the joystick to do a slow jog normally, but a fast jog when the "B" button is depressed. The speed for the slow jog is 1.0 mm/sec, and the speed for the fast jog is 2.5 mm/sec.

Possible settings:

Bit 1	Bit 2	Control Pad Motion	"B" Button Motion
on	**on**	**Slow Jog (bits 3-5 speed)**	**Fast Jog (bits 6-8 speed)**
off	on	Small Step (bits 3-5 size)	Big Step (bits 6-8 size)
on	off	Step (bits 3-5 speed, bits 6-8 size)	Repeat (bits 3-5 speed, bits 6-8 size)
off	off	Step (bits 3-5 speed, bits 6-8 size)	Jog (bits 3-5 speed, bits 6-8 size)

Bit 3	Bit 4	Bit 5	Step Size (mm)	Speed (mm/second)
on	on	on	0.00001	0.0625
off	on	on	0.0001	0.1
on	off	on	0.001	0.25
off	off	on	0.01	0.5
on	**on**	**off**	**0.1**	**1.0**
off	on	off	1.0	1.5
on	off	off	2.0	2.0
off	off	off	5.0	2.5

Bit 6	Bit 7	Bit 8	Step Size (mm)	Speed (mm/sec)
on	on	on	0.00001	0.0625
off	on	on	0.0001	0.1
on	off	on	0.001	0.25
off	off	on	0.01	0.5
on	on	off	0.1	1.0
off	on	off	1.0	1.5
on	off	off	2.0	2.0
off	**off**	**off**	**5.0**	**2.5**

NOTE: All other jumpers and switches located on the IEEE board are set at the factory and are not meant to be changed by the customer. If, for some reason, these have been changed, please contact Melles Griot technical support for the correct settings.

4.4.4 IEEE and RS-232 System Software Installation

The Nanomotion II IEEE/RS-232C system has a microprocessor included on the board that has the ability to accept commands sent over the IEEE and RS-232 interfaces. See Chapter 6, Software Commands, for a list of these commands. There is no additional software required.

LabVIEW Drivers

The following files will be included on the diskettes packed with the system. They are designed for use with LabVIEW for Windows:

Disk	File	Function
22 SFT 101	lv3gpib.zip	contains a LabVIEW library of VIs for use with LabVIEW 3.x IEEE commands
22 SFT 103	lv3visa.zip	contains a LabVIEW library of VIs for use with LabVIEW 3.x VISA commands.
22 SFT 105	lv4gpib.zip	contains a LabVIEW library of VIs for use with LabVIEW 4.x IEEE commands.
22 SFT 107	lv4visa.zip	contains a LabVIEW library of VIs for use with LabVIEW 4.x VISA commands.

4.4.5 Cables

A standard IEEE 488 cable or RS-232 cable with 9-pin connector is required for connection to a computer system. These are not supplied with the Nanomover system. If operating additional axes, the expansion board is connected between the Nanomotion II Controller Chassis by a two-meter cable. This cable is terminated on both ends with a 50-pin "D" type connector. The cable is physically keyed by the "D" connector so that it cannot be inserted incorrectly.

4.4.6 Connecting a Joystick

A joystick connected to the Nanomotion II Controller chassis with a IEEE 488.2/RS-232C interface board can be used to control the system, even without the benefit of a host computer. It is plugged into the 9-pin connector in the upper left-hand corner of the IEEE/RS232 Controller chassis.

NOTE: The joystick should be a switch-style (contact-closure) joystick or control pad. The more common PC-interfaced joysticks are analog and will not work with the joystick port.

4.4.7 Configuring Additional Axes

When more than two axes will be in use, the user has the option of setting up the system as a single GPIB address or as multiple GPIB addresses. The choice determines the equipment required.

Item	Single Address	Multiple Addresses
Master Unit (first two axes)	11 NCS 101/IEEE	11 NCS 101/IEEE
Additional Unit (per two additional axes)	11 NCS 101/IEEE	11 NCS 101/IEEE
Expansion Card	11 NIB 001	Not Required

Setup for a Single GPIB (IEEE) Address

To set up, follow the instructions in this section to:

1. Add an expansion card (11 NIB 001) to the 11 NCS 101/IEEE controller in the bottom slot of the back panel of the chassis. Make sure that the Master/Slave jumper on this board is set to Master for this chassis, and that all of the resistor SIPs on the board remain installed.
2. Remove terminating resistors from all remaining expansion cards in the 11 NCS 101 slave controllers with the exception of the last one in the chain. For example, if only one additional controller is used, no

resistors would be removed. If two additional controllers are used, resistors need to be removed from one controller, and this controller needs to be daisy chained between the other two. The Master/Slave jumper on this board should remain set to Slave.

3. Set address switches on the back of the controller at different addresses for each controller box, following the instructions in the table below. The 11 NCS 101/IEEE controller's address needs to remain at 300, the address set at the factory.
4. Daisy chain the controllers together with the supplied expansion cables. Remember to keep the controllers with terminating resistors installed at the ends of the chain.

NOTE: The joystick will only operate without computer control on the 11 NCS 101/IEEE controller box in this configuration.

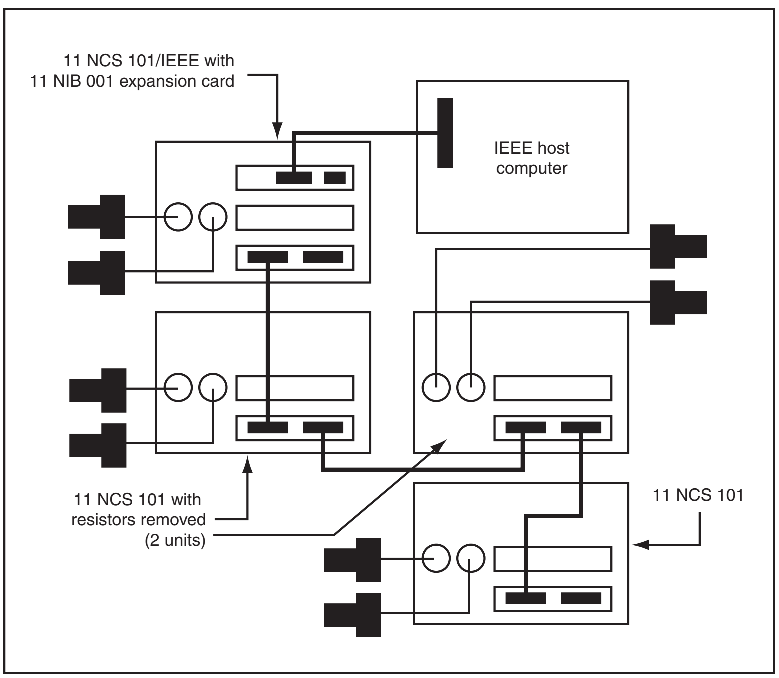


Figure 4.6 Hookup for eight axes of motion

Expansion Board Setup

When using the IEEE/RS232 System with more than two motors, the expansion board is used to daisy chain controllers together. Remove the board from its protective bag and identify the bottom slot on the Controller Chassis with the IEEE/RS-232 interface board. Remove the cover from this slot. Set the Jumper at J1 (Master/Slave jumper) to Master (see Figure 4.7 for jumper settings). Insert the board into the slot, ensuring that it is seated correctly. Screw the retaining pins onto the posts. Connect the cable to the next Controller Chassis (expansion board) for multiaxis operation.

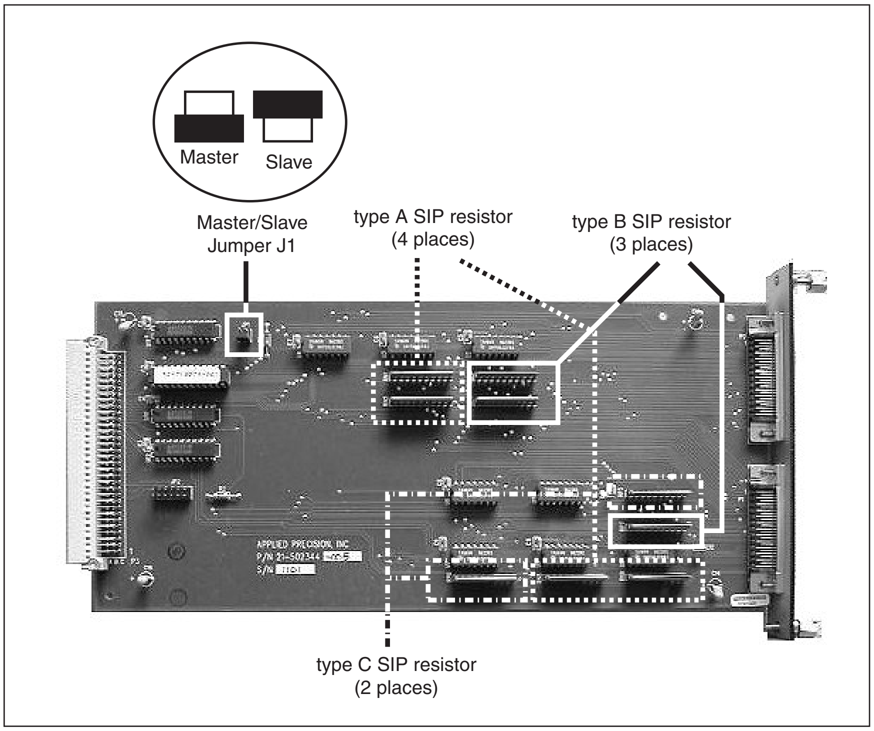


Figure 4.7 Expansion Board and Jumper Settings

WARNING: Be sure to wear a grounding strap or take appropriate precautions to avoid damaging the Nanomotion II boards with static electricity. When removing or replacing boards in the Controller Chassis be sure that the power is turned off. Removing or replacing boards with power on will cause them to be damaged.

Terminating Resistors

If more than four axes will be used at one GPIB address, the terminating resistors will need to be removed from all PC bus boards (expansion boards) except the last one in the chain.

When removing the terminating resistors, designated as Resistor SIP Types A, B, and C in Figure 4.6, it is essential that you keep the resistors together by type, as they will be required if the controller will ever be used alone or at the end of a chain.

Jumper Settings

To replace the resistors, make sure that pin 1 on the SIP goes to the end where the resistor location is marked (e.g., RN1) on the board. Pin 1 on the SIP is usually designated by a dot or a vertical line on one end of the resistor. If you have not kept the resistors, use an equivalent to the following, or contact Melles Griot customer service for a resistor set.

Resistor Type	Value	Number of Resistors	Internal Configuration	SIP Quantity on Board
Type A	330 Ω	9	Pin 1 common	4
Type B	330 Ω	5	Each isolated	3
Type C	110 Ω	5	Each isolated	2

Setting the PC Addresses

When multiple chassis are daisy chained together, each board must be assigned a unique address so that information can be exchanged without becoming lost or misdirected. The addresses are assigned using a series of 8 DIP switches, together referred to as SW1, located on the back of the Controller Chassis.

Use Figure 4.1 to determine the correct SW1 switch settings for your Nanomotion II board(s). Each board can control two Nanomovers that are referenced in the software by their motor numbers. It is recommended that the IEEE chassis remain at address 300, and that all additional units be in contiguous addresses above it.

Setup for the IBM System

Configuring multiple axes for the IBM System follows the same steps as for the single address IEEE systems but with the following exceptions:

1. The PC link card is the master of the system and therefore all chassises connected to the host computer must have the master/slave jumper, JP1, set to slave. See Figure 4.6.
2. If more than 2 axes are used (more than 1 chassis) all of the terminating resistors will need to be removed from all the expansion boards except the last one in the chain.
3. The IBM and expansion chassis come complete with the expansion board installed. No additonal boards need to be installed.

Setup for Individual GPIB Addresses

As mentioned at the beginning of this section, the user will need an additional 11 NCS 101/IEEE for each additional 2 axes.

To set up, follow the instructions given in section 4.4.3 to:

1. Set switch 2 on the GPIB board to a different GPIB address for each controller.
2. 2. Hook each controller to the computer via the IEEE address.

Each controller chassis will have to be addressed individually via the GPIB bus.

4.5 Nanomover Motor Installation

Each Nanomover includes a permanently attached three-meter cable. This cable can plug into either of the two motor connectors on the back of the Nanomotion II controller chassis. If it is necessary to operate the Nanomovers at distances greater than three meters from a controller chassis, additional extension cords are available for distances up to 35 meters.

CAUTION: Under no circumstances are improvised cables to be used as doing so may damage the system.

4.5.1 Limit Switches

Limit switches are not considered necessary, but some users may wish to install them for added safety. Nanomotion II systems are configured for user-supplied limit switches of the normally-open switch contact or opto-interrupt type. Whenever a limit switch is closed by a Nanomover, the

Nanomover will not be driven any further in that direction. See Limit Switch Circuits in Chapter 3, Specifications, for more information.

4.5.2 Attaching Nanomovers to Stages

Proper connection of a Nanomover to its stage is important for obtaining high-resolution movement. The mounting barrel of the Nanomover is a standard metric size designed for mounting in a 10 mm diameter socket within a 10 mm thick block. High quality translation stages should be used to avoid degraded performance.

To attach a Nanomover to a stage, first retract the micrometer shaft using the manual movement knob, leaving approximately 1.5 mm of the shaft exposed. It is important that about 1.5 mm of shaft is exposed, because the synchronization of position and micrometer shaft displacement registers expect this length. Remove the protective cap. The Nanomover can now be inserted in the mounting socket of the translation stage. The mounting nut should be screwed onto the shaft to hold the Nanomover in place. To further ensure against mechanical slippage, the set screw in the mounting block of the stage, if applicable, should be tightened onto the mounting barrel.

WARNING: Take care to avoid overtightening the set screw, as doing so will damage the Nanomover barrel.

Turn the manual positioning knob after tightening the mounting set screws and nut, to ensure that the micrometer shaft can turn freely. If the spindle binds or is difficult to turn, slightly loosen the setscrews and the nut until it turns freely.

CAUTION: To prevent long term wear to the stage and micrometer tip, a small drop of "High Pressure" grease should be placed on the contact point of the stage. Use of light viscosity oils should be avoided because it may flow onto the micrometer barrel and further into the Nanomover housing — potentially carrying contaminants to the micrometer threads.

(This page intensionally left blank.)

Chapter 5

Operating The Nanomotion™ II System

5.1 Operating Nanomotion™ II Using the Nanomotion™ II Control Program

With the Nanomotion II Control Program, up to 16 axis of motion can be controlled via a Windows-based application that provides powerful functionality through an easy-to-use interface.

NOTE: To use the Nanomotion II Control Program, your IBM-compatible PC must be equipped with a PC-Link card, the controller must have an installed IBM interface board (11 NIB 001), and the IEEE 488.2/RS-232C interface board, if present, must be removed.

5.1.1 Operating Conventions

The Nanomotion II Control Windows Program follows most of the standard conventions for Windows applications. It uses a menu-driven interface and windows to provide information and access to the system. Specific information about the control program is later in this section; below are some general operating conventions.

OK vs. Cancel

Two buttons can be used to close most windows. OK is used to accept any modifications that you have made to the parameters listed in that window. Cancel is used to close the window and revert to the parameter values that existed when the window was opened.

Escape key

The escape key will close the open window, reverting to the parameter values that existed when the window was opened (similar to the Cancel button).

Standard Filename Extensions

Nanomotion II relies on several standard extensions to identify files.

File Extension	File Description
.axs	Axis Configuration File
.ini	Initialization File
.exe	Executable File
.DLL	Dynamic-Link Library

5.1.2 Defined Movement Types

Six movement commands (TAB, JOG, STEP, REPEAT, ABSOLUTE, SCAN) are provided with the Nanomotion II Control Program. These commands provide a convenient means to direct Nanomotion II through nearly all useful types of motions.

Units to Move

The concept of Units to Move (or Move Length) is an important part of the Control Program. This is a user-defined variable that can be modified for each of the different types of movements, such as STEP or REPEAT, as described below.

All movements use the profile defined by the user-selected acceleration and velocity ramps. The defined movement commands available are described below;

> **ADVISORY: In all cases, depressing the pc keyboard space bar causes moving Nanomovers to stop immediately, as discussed in Stopping Movement.**

Tab

TAB causes the selected Nanomover(s) to move to the closest tab position. A plus TAB key moves to the closest tab position to the right, and a minus TAB key moves to the closest tab position to the left. (Tab requires user definition in the tab position "Setup Windows" for tab locations.)

Jog

JOG is used to quickly position the selected Nanomover(s) anywhere in its range of motion. The JOG feature operates in two modes. If a JOG motion key is pressed and then released quickly, the Nanomover will move the Units to Move amount. For example, if Units to Move is set to 1 mm and a +JOG key is pressed and released quickly, the designated Nanomover will move by

+1 mm. If Units to Move is set to 50 nanometers, the movement will be +50 nanometers. However, if the JOG key is pressed and held down, the Nanomover will continue to move in a free run mode as long as the key is held down, i.e., not necessarily in multiples of Units to Move. Releasing the key will halt movement.

Step

STEP is used to move the selected Nanomover(s) by the amount of the “Units to Move” variable. It is similar in function to the JOG command, but will not repeat unless re-selected (that is — unlike JOG, holding down the STEP key does not produce “FREE RUN” motion). Hence, STEP is useful for making small, precise movements. STEP will not cause a Nanomover to move outside the left or right stops.

Repeat

REPEAT is a combination of STEP and JOG. REPEAT is used to move the selected Nanomover(s) in specific increments (like STEP), but can also be used to make several STEP moves at a time. If a REPEAT key is pressed and then released quickly, the Nanomover will move the Units to Move amount (same as for STEP key). However, if a REPEAT key is pressed and held down, the Nanomover will begin to make a rapid series of moves of the size Units to Move, and will continue to make these moves as long as the key is held down. Releasing the key will stop the movement. Repeat will not move outside the left or right stops.

Absolute

ABSOLUTE is used to move the selected Nanomover(s) to either the home or zero position (see section 2.8). Pressing a plus ABSOLUTE key causes the Nanomover to move to the Home position, while pressing a minus ABSOLUTE key causes the Nanomover to move to the zero position. This command is useful for restoring the Nanomover to a convenient starting point.

Scan

SCAN causes the selected Nanomover(s) to go back and forth between the closest Tabs. When a plus SCAN key is pressed, the Nanomover will go to the closest Tab that is located in a positive direction. It will then scan back and forth between that Tab and the closest negative (left) Tab. If the Nanomover is already positioned at a Tab location when the plus SCAN key is pressed, it will scan between the current Tab and the closest left Tab. The Nanomover will continue scanning until the minus SCAN key is pressed. When SCAN is

stopped using the minus scan key, the Nanomover will always stop at the right Tab position.

When a minus SCAN key is pressed, (to initiate scan) all directions described in the preceding paragraph are reversed.

5.1.3 Starting the Nanomotion II Control Program

Start the Control Program by double-clicking on the Nanomotion II icon that you created in the installation procedure. Alternatively, you can start the program by using the Run feature of the File Manager.

5.1.4 Operating the Nanomotion II Control Program

Nanomotion II System Window

Figure 5.1 below shows a typical system window for Nanomotion II. The various menu headings are visible under the title bar. Each of the other parameters is explained below.

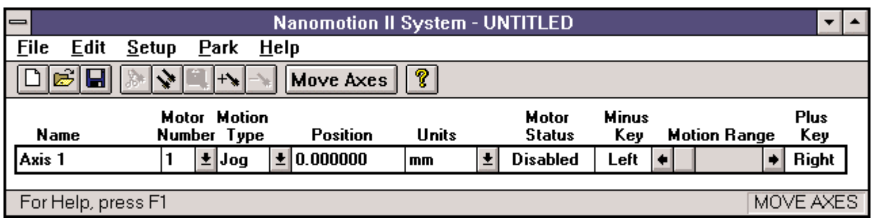


Figure 5.1 Nanomotion II System window

Name

Identifies the name of each axis. The name you select should be something that helps you identify the function of the axis, i.e., X-Translation.

Motor Number

Identifies the Nanomover being controlled.

Motion Type

One of the 6 available types of movement available with the Control Program (tab, jog, step, repeat, absolute, scan).

Position

Absolute location of the Nanomover.

Units

Units currently selected to measure distance. See **Units Window** for more information.

Motor Status

Information about the status of the Nanomover. The keywords that may appear in this box are described below.

Motor Status	Meaning
Normal	Normal stationary status
Left Limit	Left limit switch is active
Right Limit	Right limit switch is active
Left Stop	Located at left absolute stop
Right Stop	Located at right absolute stop
Moving	Moving
Parked	Parked

Motion Range

A mouse or keyboard-operated movement control. Clicking on the slider box, or pressing the indicated keys under Minus Key or Plus Key, will cause the Nanomover to move.

Pull Down Menus

File Menu

The File Menu contains standard MS Windows commands for control of configuration files. These files contain descriptions of Nanomotion II configurations, such as number of axes and operating parameters.

New

Create a new file. A new Nanomotion II configuration file, with default values for each parameter, will be created. Use this command when setting up a new system.

Open

Open an existing configuration file.

Save

Save a configuration file using the same name it was opened with.

If the file has not been saved previously (i.e., it was created with the New command), you will be prompted for a filename.

Save As

Save a file, prompting for a name. A new name may be entered, or the old name may be used. If you use the old name, the previous file will be overwritten and destroyed.

Exit

Quit the Nanomotion II Control Program.

Edit Menu

The Edit Menu contains commands to modify the parameters that describe the movement axes.

Add Axis

Create a new axis, complete with default values for all parameters.

Copy Axis

Copy the highlighted axis to a memory buffer.

Cut Axis

Copy the highlighted axis to a memory buffer, then delete the highlighted axis.

Paste Axis

Copy the highlighted axis from a memory buffer to the **System Window**.

Edit Axis

Modify the parameter values for the selected axis.

Delete Axis

Remove the selected axis. This command, once issued, cannot be undone.

Setup Menu

The Setup Menu provides access to the window that controls the configuration of the Nanomotion II electronics hardware, including PC card addresses.

Hardware

For the software to properly use the Nanomotion II board, it must be informed about the location of the board and each of the axes

being used. See **Hardware Configuration Window** for more information.

Units

Adding mechanical hardware to the Nanomovers controlled by the Nanomotion II system could give a setup that would be more conveniently controlled in units other than those pre-defined by the system. See the **Units Setup Window** for more information.

Park Menu

The Park Menu contains the commands to Park and Unpark the Nanomovers.

Park Nanomovers

Park the Nanomovers.

Unpark Nanomovers

Unpark the Nanomovers.

Help Menu

The Help Menu provides access to the on-line help system.

Nanomotion Help

A comprehensive help utility to explain the Nanomotion II system and its use.

Using Help

Help information on how to work with Windows graphical interfaces.

About Nanomotion II

Information about the instrument.

Motor Parameters Window

The Motor Parameters window is available for each of the axes used in Nanomotion II. It contains information summarized in the System Window, as well as several additional parameters. Access the window by clicking on Edit Axis under the Edit menu, or by double-clicking on an axis in the System Window. Each feature in the window is described below, or under the **System Window**.

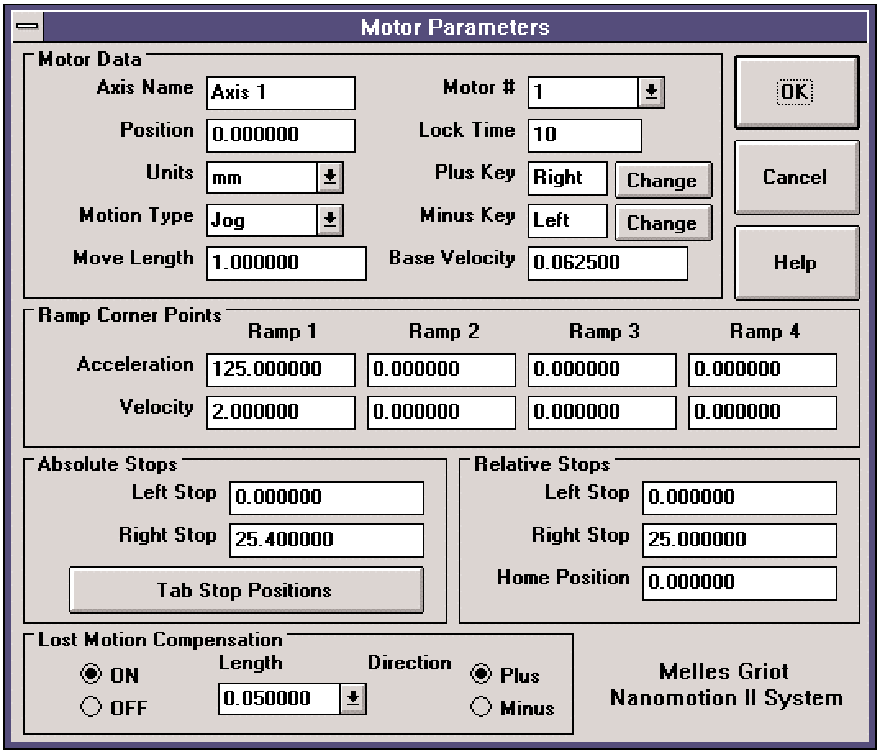


Figure 5.2 Motor parameters window

Motor Data

Axis Name

The name of the selected axis.

Position

The current location of the Nanomover.

Units

A pull-down menu to control which measurement units are currently being used.

Motion Type

A pull-down menu to control which type of motion is currently active.

Move Length (Units to Move)
Movement step-size.

Motor
Identifies the selected Nanomover.

Lock Time
Length of time, in milliseconds, that the Nanomover is to maintain a high-torque condition after each movement.

Plus Key
Key dedicated to moving the Nanomover in the positive direction, using the selected movement type.

Minus Key
Key dedicated to moving the Nanomover in the negative direction, using the selected movement type.

Base Velocity
Speed at which the Nanomover will begin each move unless otherwise instructed by an acceleration profile.

Ramp Corner Points
The acceleration and velocity for each of the 4 regions of the acceleration profile.

Absolute Stops
The left and right locations, with respect to the zero position, beyond which the Nanomover will not move.

Relative Stops
The left and right locations, with respect to the home position, beyond which the Nanomover will not move.

Lost Motion Control (LMC)

On/Off
Controls whether the lost motion compensation feature, which improves resolution and accuracy, is active. See **Lost Motion Compensation in Theory of Nanomotion II** for more information.

Length

Magnitude of the LMC, if it is turned on.

Direction

Direction of the LMC, if it is turned on.

Edit Tab Positions Window

The Edit Tab Positions Window lists the position of all 16 available tab positions. See the Defined Movement Types for information about how the tab positions are used by the Nanomotion II Control Windows program.

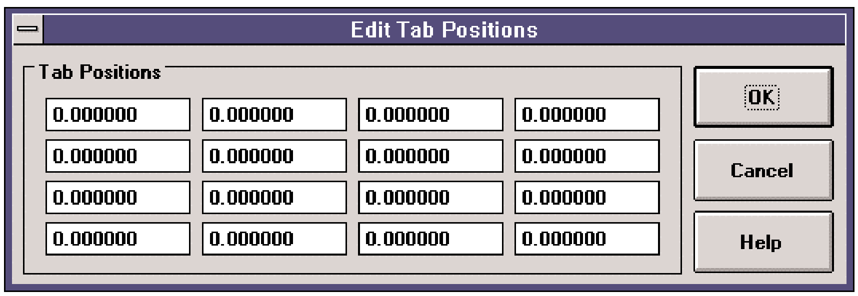


Figure 5.3 Tab positions window

Set Base Address [Hardware Configuration] Window

The Set Base Address Window describes the addresses necessary for the software to communicate with the Nanomotion II electronics hardware.

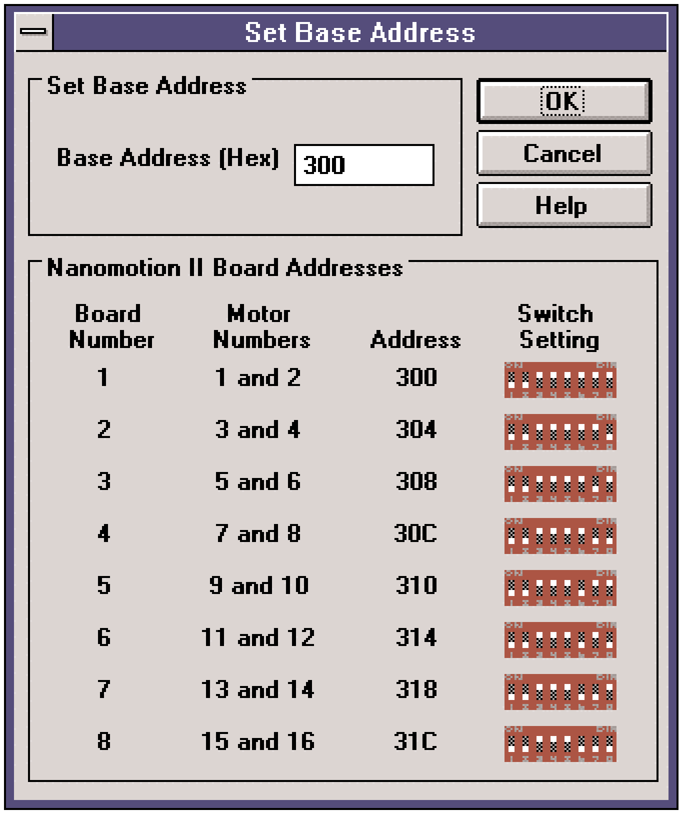


Figure 5.4 Set base address window

Base Address

Determines the address that the PC will use to communicate with the installed PC-Link board.

Board Addresses

Determines the addresses that the PC will use to communicate with each of the installed Nanomovers.

Units Window

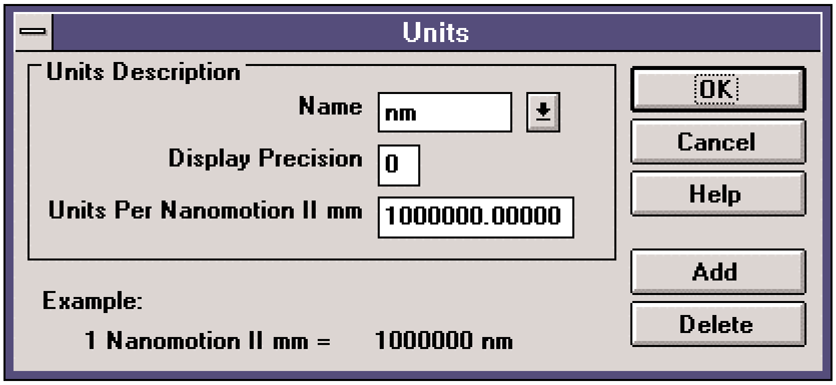


Figure 5.5 Units Window

The Units Window permits custom, user-created, measurement units to be entered into The Nanomotion II control program. Custom units can be useful for applications with unique requirements. For example, a Nanomover that rotates a cylinder might be better described in terms of radians, where the exact amount of movement per step is determined and entered into the program. The standard measurement units, such as mm (millimeters), cannot be modified.

5.2 Operating with LabVIEW® Drivers

NOTE: To operate with LabVIEW drivers, the controller must be equipped with a IEEE 488.2/RS-232C expansion board that has been properly configured.

5.2.1 List of Sub-Virtual Instruments (VIs) Supplied

Each system that is equipped with the IEEE 488.2/RS-232C expansion board ships with LabVIEW® drivers for LabVIEW 3.x and 4.x. These drivers consist of a library of sub-VIs and an example VI. The complete list is as follows:

MG Nanomotion II Example.VI

MG Nanomotion II Check for Motor.VI

MG Nanomotion II Close.VI

MG Nanomotion II Get Range/Position.VI

MG Nanomotion II Initialize.VI

MG Nanomotion II Lost Motion Comp.VI

MG Nanomotion II Motor Parameters Global.VI

MG Nanomotion II Motor Parameters.VI

MG Nanomotion II Move.VI

MG Nanomotion II Park/Unpark.VI

MG Nanomotion II Query SBR.VI

MG Nanomotion II Query SESR.VI

MG Nanomotion II Read Limit Switches.VI

MG Nanomotion II Read Ramps.VI

MG Nanomotion II Read Status.VI

MG Nanomotion II Remote/Local.VI

MG Nanomotion II Reset Motor.VI

MG Nanomotion II Reset.VI

MG Nanomotion II Revision Query.VI

MG Nanomotion II Self-Test.VI

MG Nanomotion II Set Absolute Stops.VI

MG Nanomotion II Set Back Porch.VI

MG Nanomotion II Set Base Velocity.VI

MG Nanomotion II Set Current Position.VI

MG Nanomotion II Set Current.VI

MG Nanomotion II Set Gain/Steps.VI

MG Nanomotion II Set Limit Switches.VI

MG Nanomotion II Set Position Lock Time.VI

MG Nanomotion II Set Ramp Values.VI

MG Nanomotion II Set Units.VI

MG Nanomotion II Set-Query SESER.VI

MG Nanomotion II Set-Query SRER.VI

MG Nanomotion II Stop Motion.VI

Complete documentation is available for all of these VIs using the LabVIEW on-line HELP function. See the Command Reference section in Chapter 6, Software Commands for details on each command issued within the IV.

5.2.2 The Example VI

Basic Operation of the Nanomotion II system can be done via the MG Nanomotion II Example VI included in the library. The values in this VI are set to the defaults as discussed in this manual.

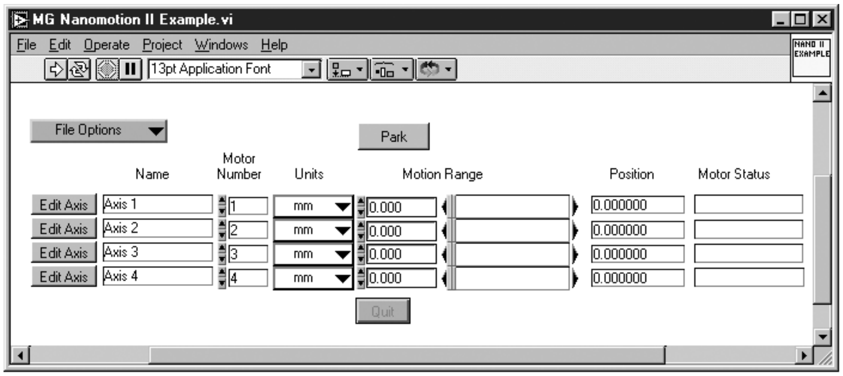


Figure 5.6 LabVIEW Example VI display

Running the Example VI allows the motors to be moved either by inputting a number location or by moving the motion range indicator. The motors can also be parked and unparked.

By scrolling up from the screen as shown above, the user can change the GPIB address of their unit.

When the VI is started, it will check for multiple axes and will shade out axes it does not find, as shown above.

NOTE: if the IEEE chassis does not have the address 300 on the motor control board, the VI may not find any axes.

File Options allows changes (final motor position, etc.) to be saved or retrieved in a unique file.

Park will either park or unpark all motors.

Edit Axis brings up another screen (see Figure 5.7) where additional configuration of the motor can be made.

Quit will stop running the VI.

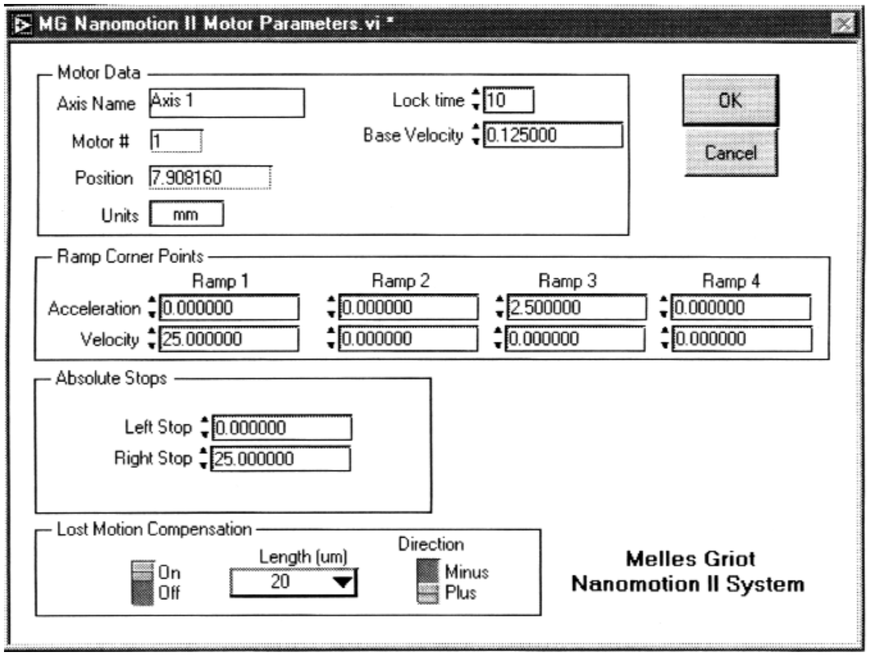


Figure 5.7 Edit Axis screen

CAUTION: There are no safeguards in these drivers to keep you from putting in numbers that will not work. Numbers out of range should not damage the Nanomotion II system unless they are used repeatedly. Please refer to this manual for correct settings if the response is not as expected, or contact Melles Griot customer support before continuing.

5.3 Operating with the RS-232C Interface

5.3.1 Using the Windows HyperTerminal Program

The Windows terminal screen shown in Figure 5.8 below shows the default settings required to communicate with the Nanomotion II RS-232 interface. The characters at the left of the screen are interactive commands given to the system.

NOTE: The baud rate can be changed by configuring the dip switches on Switch 1 as shown in Section 4.4.

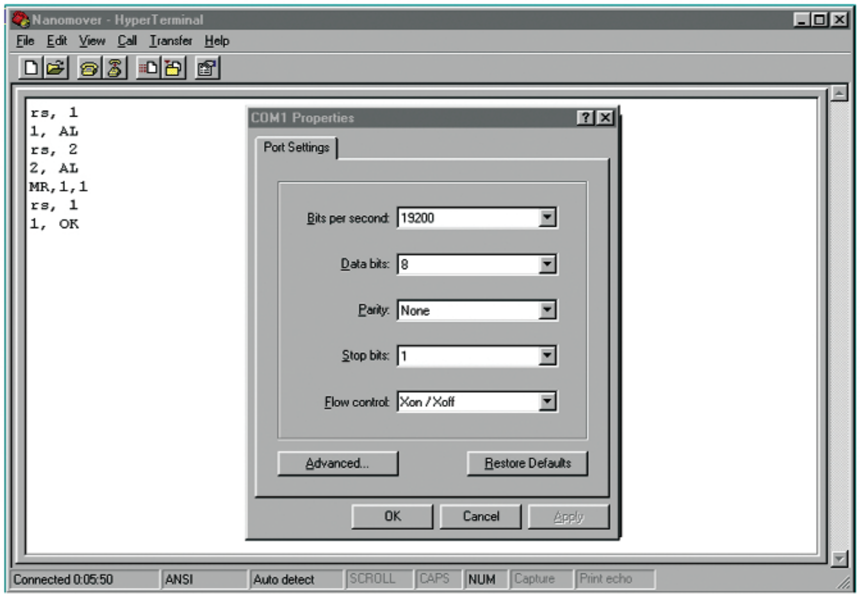


Figure 5.8 Windows HyperTerminal Screen

5.4 Serial Port Programming Hints

Each computer has different requirements for using its RS-232C (serial) interface ports. You must consult the documentation for specific information about using the ports to their full capability.

Serial ports are notorious for being simple devices that are difficult to operate and debug, because of the uncertainty that can exist over what is actually being received by the intended target device.

Any unspecified parameters will be assigned default values. The BREAK command should be sent and the ACK response received at the beginning of each program, in order to synchronize the serial systems. In addition, the RESET command should be sent to clear any random residuals, thereby resetting the system to its default values.

It is good programming practice to use timeouts and an error handler when sending or receiving information. A timeout of a few hundred milliseconds will prevent the computer from locking up if the serial port becomes incommunicative, while the error handler can indicate where any problems occur.

5.4.1 Software Example for Serial Interface

The following is an example of programming for the serial port. This was developed using a communications library called ENCOM, which is a shareware product.

```
#include <stdlib.h>
#include <stdio.h>
#include "encom.h"

PORT Port;
/**
**/
/****************************************************
******
****/
/**
**/
/****************************************************
******
/**
**/
void SendString (char *String)
{
int c;
int a,j;
int i=0;
while (*(String + I) !=(char) NULL)
  {
if ( (a=com_getc(&Port)) !=-1) putchar(a);
c=(int)*(String + i)
putchar©;
com_putc(c, &Port);
i++;
  }
}
void main()
{
char out[ 50] ;     /* Constant Declarations */
int port = COM2;
int end_flag = 0, c, echo = 0;
int motor = 1;
int lmc = 0;
```

```
float lstop = -100.0;
float rstop = 100.0;
float bvel = .125;
float vell = 1.0;
float accell = 5.0;
float move2 = 0.0;
long baud = 9600L;

if(com_port_create(port,baud,'N',8,1,
2048,2048,&Port)<0)
printf("cannot initialize COM port");
init_clock(0´3333);
init_ctric_hdlr();
com_232_ctrl(ON, DTR | RTS | OUT2, &Port);
sprintf(out, "WAL, %d, %f \r\n", motor, lstop);
SendString(out);
sprintf(out, "WAR, %d, %f \r\n", motor, rstop);
SendString(out);
sprintf(out, "WLM, %d, %f \r\n", motor, lmc);
SendString(out);
sprintf(out, "WB, %d, %f \r\n", motor, bvel);
SendString(out);
sprintf(out, "WVl, %d, %f \r\n", motor, vell);
SendString(out);
sprintf(out, "WAl, %d, %f \r\n", motor, accell);
SendString(out);
sprintf(out, "MR, %d, %f \r\n", motor, move);
SendString(out);
Motor = 2;
sprintf(out, "MR, %d, %f \r\n", motor, move);
SendString(out);
sprintf(out, "MA, %d, %f \r\n", motor, move2);
SendString(out);
Motor=1;
sprintf(out, "MA, %d, %f \r\n", motor, move2);
SendString(out);

End_clock();
End_ctrlc_hdlr();
Com_port_destroy(&Port);
}
```

end of example

Chapter 6

Programming Nanomotion™ II

For some applications, the power and flexibility of the Nanomotion II Control Program may not be appropriate. Instead, users may prefer to control the Nanomotion II system using a custom application. Recognizing this need, several programming options have been made available. First, users can use a comprehensive DLL (dynamic-link library) to construct Windows-based applications in nearly any Windows development environment. Second, an MS-DOS based application can be created using C and a linkable library of Nanomotion II functions. Third, the system can be controlled with LabVIEW for Windows. Note that the provided LabVIEW drivers will only work with the 11 NCS 101/IEEE system. And finally, the parameters that control movement can be accessed directly by users controlling the IEEE 488.2 or RS-232C port of their computer, using the SCPI-compatible command set.

All of these options are explained in the sections that follow.

6.1 IEEE 488.2 and RS-232C Commands

6.1.1 Available Commands Unique to IEEE 488.2

*CLS	*IDN?	*SRE?
*ESE	*OPC?	*STB?
ESE?	*OPC?	*TST
ESR?		

6.1.2 Available Commands Unique to RS-232

BREAK

6.1.3 Commands Available to Both IEEE 488.2 & RS-232C

*LOC	RAR	RLT	UA	WLIME
*REM	RB	RP	WA	WLM
*RST	RBPE	RRES	WAL	WLT
*WAI	RDIO	RRINTLK	WAR	WP
DENCC	REBP	RS	WB	WSCUR
DSTP	REBPE	RSCUR	WBPE	WSTP
LF	RENCC	RSTENC	WCURE	WU
MA	RENCE	RSTP	WEBP	WUSTP
MON	RES	RU	WEBPE	WV
MR	RGAIN	RUSTP	WENCC	
PA	RHCUR	RV	WENCE	
RA	RLIME	RVER	WGAIN	
RAL	RLM	S	WHCUR	

6.1.4 Interface Status Registers

Standard Event Status Register (SESR)

Shows eight types of events that can occur within the Nanomotion II system. Use the *ESR? query to read the SESR register. Reading the register clears the bits of the register so that the register can accumulate information about new events.

Bits:	7	6	5	4	3	2	1	0
Meanings:	PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

PON	Power On	Shows that the Nanomotion II system was powered on.
URQ	User Request	Shows that a local control was pressed (the Joystick).
CME	Command Error	Shows that an error occurred while the system was parsing a command.
EXE	Execution Error	Shows that the system detected an error while executing a command.

DDE	Device Error	Shows that a device (Nanomotion hardware) error occurred.
QYE	Query Error	Not Used
RQC	Request Control	Not Used
OPC	Operation Complete	Shows that the operation is complete.

Status Byte Register (SBR)

The SBR shows whether output is available in the Output Queue, whether the Nanomotion II system is requesting service, and whether the SESR has recorded any events.

To read the contents of the SBR, use a serial poll or the *STB? Query. The bits in the SBR are set and cleared depending on the contents of the SESR, the ESR, and the Output Queue. When a serial poll is used to obtain the SBR, bit six is the RQS bit. When the *STB? Query is used to get the SBR, bit 6 is the MSS bit. The Status Byte Register is not reset when read.

Bits:	7	6	5	4	3	2	1	0
Meanings:	—	RQS MSS	ESB	MAV	—	—	—	—

RQS	Request Service	From a serial poll. The system requests service from the GPIB CIC.
MSS	Master Status Summary	From a *STB? Query. Summarizes the ESB and MAV bits in the SBR.
ESB	Event Status Bit	When set to 1, show that one or more of the enabled bits in the SESR have been set to 1.
MAV	Message Available	Set to 1 when data is available in the output queue.

Service Request Enable Register (SRER)

The SRER controls which bits in the SBR generate a Service Request and are summarized by the Master Status Summary (MSS) bit. The RQS bit remains set at one until either the Status Byte Register is read with a serial poll, or the MSS bit changes back to a zero.

Bits:	7	6	5	4	3	2	1	0
Meanings:	-	-	ESB	MAV	-	-	-	-

ESB	Event Status Bit	Shows that status is enabled and present in the SESR.
MAV	Message Available	Shows that output is available in the Output Queue.

6.1.5 Command Syntax

The general syntax of the command set is "Command, Motor Number, Data." The command, motor number, and data are ASCII strings separated by a comma (,). Spaces are simply ignored when encountered and may be used to make the source code more readable if desired. Data may include decimal points and a plus or minus sign, as required. When using the RS-232 interface, a carriage return (CR) causes the command to be executed. Line feed (LF) is ignored if present.

Data is returned from the Nanomover system with the syntax "Motor Number, Data." Via the serial interface, data is terminated with CR or CRLF as specified by the host system. Default is CRLF, which is the designation used in all of the examples.

Commands are always expressed with reference to the host computer. A write instruction indicates writing to the Nanomotion chassis, a read indicates reading from the Nanomotion chassis. Data requested by a read instruction will cause the Nanomotion chassis to respond immediately with the appropriate data.

If due care is given to calling safe absolute right and left stops, the Nanomover system will protect itself against accidental damage by improper commands. The Nanomover system will NEVER move beyond an absolute left or absolute right stop, regardless of the command given (except for certain instances with LMC enabled (see Chapter 2, Introduction to Nanomotion II).

The actual command syntax is described in the following. Letters given in CAPITALS must be sent exactly as shown. Words enclosed in angle brackets <> describe variables which will be represented by numeric strings. The angle brackets are for clarity in this manual. Do not type the brackets in the actual program.

With IEEE commands, several commands can be sent in one string, with each

command separated by a semicolon. For serial interface commands, each command must be terminated with a CR or a CRLF. In the examples given, the values are described in terms of units. The units for each Nanomover may be specified independently, and all values are interpreted in terms of the specified units. The words <mtr> refer to individual motors numbered one through sixteen.

Example: MR,2,-12.34

This command will cause Nanomover #2 to move to the left 12.34 units. If units are specified as millimeters, the shaft of Nanomover #2 will retract 12.34 mm.

6.1.6 Command Details in Alphabetical Order

*CLS

Command: Clear Interface Status Command

Parameters: None

Description: Clears the interface status data structures:
the Standard Event Status Register (SESR)
the Status Byte Register (SBR), except the MAV bit

Example: `*CLS`

Clears the interface status data structures.

*ESE,<on/off bits>

Command: Standard Event Status Enable Command

Parameters: <on/off bits> — The decimal sum of the bits to be set (1) in the register

Description: Set or query the bits in the Event Status Enable Register (ESER). The ESER prevents events from being reported to the Status Byte Register (SBR).

The value is determined as a decimal integer corresponding to the sum of the bit values from the register. For example, if only the least significant bit in the register (OPC) is to be turned off, the value should be "254" (0FE Hex). Only the most significant bit (PON) turned on would be "128" (080 Hex), etc.

Example: `*ESE,0`

Sets the Event Status Enable Register (ESER) to all zeros (clears the register).

*ESE?

Command: Standard Event Status Enable Command

Parameters: None

Description: Queries the bits in the Event Status Enable Register (ESER). The ESER prevents events from being reported to the Status Byte Register (SBR).

The value is returned as a decimal integer corresponding to the sum of the bit values from the register. For example, if only the least significant bit in the register (OPC) is turned off, a read of the port would return a "254" (0FE Hex). Only the most significant bit (PON) turned on would return "128" (080 Hex), etc.

Example: `*ESE?`

Queries the Event Status Enable Register (ESER). If only the PON bit is active in the register, the interface board would return the ASCII string 128 CR(LF).

*ESR?

Command: Standard Event Status Register Query

Parameters: None

Description: Query the contents of the Event Status Register (SESR). *ESR? also clears the register (reading the register clears it).

The value is returned as a decimal integer corresponding to the sum of the bit values from the register. For example, if only the least significant bit in the register (OPC) is turned off, a read of the port would return a "254" (0FE Hex). Only the most significant bit (PON) turned on would return "128" (080 Hex), etc.

Example: `*ESR?`

Queries the Event Status Enable Register (ESER). If only the PON bit is active in the register, the interface board would return the ASCII string 128 CR(LF).

*IDN?

Command: Identification Query

Parameters: None

Description:	Query the identifying information about the instrument and its firmware.
Example:	`*IDN?`
	Returns the Nanomotion II identification code. The interface board might return the ASCII string Melles Griot Nanomotion II GPIB/RS232-C Interface — v2.0 CR(LF).

*LOC

Command:	Local Operation Command
Parameters:	None
Description:	Enable local instrument operation. For the Nanomotion II system, the GPIB and RS-232C interfaces will begin to ignore all commands except 'Remote Operation Enable' (*REM). And will permit joystick control of axes
Example:	`*LOC`
	Disables interface control of the Nanomotion II system (except for the *REM command). And enables Joystick control.

*OPC

Command:	Operation Complete
Parameters:	None
Description:	Sets the operation complete (OPC) bit message in the Standard Event Status Register (SESR) when all pending operations are finished.
Example:	`*OPC`
	Commands the interface board to set the OPC bit in the Standard Event Status Register (SESR) when all pending operations are finished.

*OPC?

Command:	Operation Complete Query
Parameters:	None
Description:	Places the ASCII character "1" into the output queue when all pending commands are complete. The *OPC? response is not available to read until all pending operations finish.

Example: `*OPC?`

Returns and ASCII “1” if all pending commands are complete.

*REM

Command: Remote Operation Command

Parameters: None

Description: Enable remote instrument operation. For the Nanomotion II system, the GPIB and RS-232C interfaces will begin to recognize all normal commands. Joystick control of axes will no longer be permitted.

Example: `*REM`

Enables interface control of the Nanomotion II system and disables Joystick control.

*RST

Command: Reset Command

Parameters: None

Description: Returns the system to a known state, but does not purge any aliases or stored settings. For the Nanomotion II system, this command stops all Nanomovers and clears the communications interface.

This is slightly different from the RES command, which does change the motor settings (sets them back to a default state).

Example: `*RST`

Resets the Nanomotion II system without changing any aliases or stored settings.

*SRE,<on/off bits>

Command: Service Request Enable

Parameters: <on/off bits> — The decimal sum of the bits to be set (1) in the register

Description: Sets the bits in the Service Request Enable Register (SRER).

The value is determined as a decimal integer corresponding to the sum of the bit values from the register.

Example: *SRE,0

Sets the Service Request Enable Register (SRER) to all zeros (clears the register).

*SRE?

Command: Service Request Enable

Parameters: None

Description: Queries the bits in the Service Request Enable Register (SRER).

The value is returned as a decimal integer corresponding to the sum of the bit values from the register.

Example: *SRE?

Returns the value in the Service Request Enable Register (SRER).

*STB?

Command: Read Status Byte Query

Parameters: None

Description: Queries the contents of the Status Byte Register (SBR).

The value is returned as a decimal integer corresponding to the sum of the bit values from the register.

Example: *STB?

Returns the value in the Status Byte Register (SBR).

*TST?

Command: Interface Self-Test Query

Parameters: None

Description: Tests the communications interface. The command does not affect any Nanomovers, but always returns an ASCII "1".

Example: *TST?

Tests the communications interface.The interface card will respond with the ASCII string 1 CR(LF).

*WAI

Command: Wait to Continue Command

Parameters: None

Description: Prevents the Nanomotion II system from executing further commands or queries until all pending operations finish. This command allows you to synchronize the operation of the system with your application program.

Example: `*WAI`

Forces the Nanomotion II system to wait until all previous commands have been finished before continuing. That is: if commands — <commands sequence #1> *WAI <command sequence #2> — were sent to the interface, the *WAI command will make sure that all the commands of <sequence #1> have completed before continuing to <sequence #2>.

<break> -or- BREAK

Command: Reset communications link

Parameters: None

Description: Resets the serial link. Sending ASCII code ETX (control-C or 03 Hex) will cause the serial card to terminate any serial command being processed and begin searching for the first byte of a new command. The serial card will respond with the ASCII code ACK (control-F or 06 Hex) to indicate it has received and responded to the break.

In addition to the soft <break> command of ASCII ETX, a hard line break is also supported. The system will respond with an ASCII ACK when it detects either a soft or hard <break>. The hard <break> is implemented in hardware. It may be able to regain control of the system if the soft <break> is unable to do so.

Example: `<break>` (ASCII ETX character)

Resets the Serial link.

`BREAK`

Resets the serial link.

DENCC,<mtr>

Command: Determine encoder counts

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Move the motor to determine the number of encoder

	counts for each motor cardinal step. The motor will move a small distance (4 motor cardinal steps) in each direction. If the command is successful, the number of motor steps will be returned in the format "<mtr>, <ratio>".
Example:	`DENCC,1`
	This command returns
	`1, 4 CRLF`
	for a standard Nanomover (400 motor steps per revolution) with a 1600 count per revolution encoder

Note: All the encoder functions in the Nanomotion II libraries require external hardware (an encoder with electrical interface) to work properly.

DSTP,<mtr>

Command:	Determine motor steps
Parameters:	<mtr> — Nanomover Number, 1 through 16
Description:	Move the motor to determine the number of motor steps per revolution (assuming a 1600 count per revolution encoder). The motor will move a small distance (4 motor cardinal steps) in each direction. If the command is successful, the number of motor steps will be returned in the format "<mtr>, <steps>".
Example:	`DSTP, 1`
	This command would return
	`1, 400 CRLF`
	for a standard Nanomover (400 motor steps per revolution) with the appropriate encoder.

Note: All the encoder functions in the Nanomotion II libraries require external hardware (an encoder with electrical interface) to work properly.

LF,<on/off>

Command:	Line Feed Enable or Disable
Parameters:	<on/off> — Enable string, "ON" or "OFF"
Description:	Determines whether or not to send a line feed (LF) as part of the data terminating character. The serial ports on some computers require CRLF, whereas others only require CR — consult your computer manual. If LF is set OFF, all data

is terminated with CR. If LF is set to ON, all data is terminated with CRLF. The default system status is ON. This command affects all Nanomovers in the system.

Example: `LF,OFF`

This command changes the terminating character to CR alone (rather than the default value of CRLF).

MA,<mtr>,<dest>

Command: Move absolute

Parameters: <mtr> — Nanomover Number, 1 through 16

<dest> — Destination for move

Description: Move to a given absolute position. The Nanomover will move to the position specified, regardless of the current position. The necessary move may be in a plus or minus direction. If the position exceeds an absolute stop, the Nanomover will only move to the stop position and will not go further.

Example: `MA, 1, 1.0`

This command would cause the Nanomover to move to the 1.0 units position.

MON,<mtr>,<on/off>

Command: Move Monitor

Parameters: <mtr> — Nanomover Number, 1 through 16

<on/off> — Enable string, "ON" or "OFF"

Description: Sets the monitor function ON or OFF (the default status is OFF). This is convenient for use in interrupt driven systems to indicate the end of movement and the need for further instructions. If the monitor function is set to ON, it is activated whenever a MR (move relative) or MA (move absolute) command is issued for the assigned Nanomover number. When the monitor function is activated, the GPIB / RS-232C card will send the motor number to the system host computer when that Nanomover has stopped. This will indicate that the Nanomover whose number is sent is ready to make another move.

Example: `MON,5,ON`

Sets the monitor function to ON for Nanomover #5. Now

whenever Nanomover #5 stops after a move, a '5,MON' string will be sent to the system host computer.

MR,<mtr>,<dist>

Command: Move relative

Parameters: <mtr> — Nanomover Number, 1 through 16

<dist> — Distance for move

Description: Move relative to the current position. The Nanomover will move the distance specified relative to its current position. If the distance is positive, the Nanomover will extend by the specified distance. If the distance is negative, the Nanomover will retract. If the move exceeds an absolute stop, the Nanomover will move only to the stop position and will not go further.

Example:

```
MR, 1, -1.0
```

This command would cause the Nanomover to move 1.0 unit in the negative direction. If the Units are currently millimeters, the Nanomover will retract 1.0 mm.

PA -or- P,<mtr>

Command: Park all or Park motor

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Parks the specified Nanomovers. The command PA parks all Nanomovers. P,<mtr> parks only the specified Nanomover. In general, it is a good practice to always park the Nanomovers before turning the system off. This will avoid the inconvenience of having to reset the synchronization on power-up. The current positions and the park offset information are stored on the interface card in EEPROM and will be retained when power is removed from the system. If a motor is specified which is currently parked, this command will be ignored.

Example:

```
P,2
```

Parks Nanomover #2. The current position and the offset necessary to unpark the Nanomover will be stored on the interface card.

```
PA
```

Parks all Nanomovers. Again, the information necessary to unpark each Nanomover is stored on the interface card.

RA<ramp>,<mtr>

Command: Read acceleration

Parameters: <ramp> — Ramp Segment Number, 1 through 4

<mtr> — Nanomover Number, 1 through 16

Description: Reads the acceleration value in the current units (per second per second) for the specified velocity/acceleration ramp segment for the specified Nanomover. The <ramp> parameter specifies ramp segment 1, 2, 3, or 4.

Example: `RA1,2`

Retrieves the acceleration value for velocity/acceleration ramp #1 for Nanomover #2. If the acceleration value is 100.0 mm/s/s, the interface card will respond with the ASCII string

`2,100.00000 CR(LF)`

RAL,<mtr>

Command: Read absolute left stop

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Reads the position of the absolute left stop for the specified Nanomover in the current units.

Example: `RAL,1`

Retrieves the absolute left stop position for Nanomover #1. If the absolute left stop value is 0.0, the interface card will respond with the ASCII string

`1,0.00000 CRLF`

RAR,<mtr>

Command: Read absolute right stop

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Reads the position of the absolute right stop for the specified Nanomover in the current units.

Example: `RAR,1`

Retrieves the absolute right stop position for Nanomover #1. If the absolute right stop value is 0.0, the interface card will respond with the ASCII string

`1,0.00000 CRLF`

RB,<mtr>

Command:	Read base velocity
Parameters:	<mtr> — Nanomover Number, 1 through 16
Description:	Reads the base velocity for the specified Nanomover in the current units (per second).
Example:	`RB,1`
	Retrieves the base velocity value for Nanomover #1. If the base velocity value is 0.125 mm/second and the units are millimeters, the interface card will respond with the ASCII string
	`1,0.12500 CRLF`

RBPE,<mtr>

Command:	Read backporch enabled
Parameters:	<mtr> — Nanomover Number, 1 through 16
Description:	Reads the whether the backporch is enabled or disabled for the specified Nanomover. The final motor position for a move is the same whether the backporch is enabled or disabled. The default for Nanomotion II is backporch enabled and most system users should have no reason to change it.
Example:	`RBPE,2`
	Retrieves the backporch enabled value for Nanomover #2. If the backporch is enabled for the Nanomover, the interface card will respond with the ASCII string
	`2,ON CRLF`

RDIO,<mtr>

Command:	Read digital input (joystick)
Parameters:	<mtr> — Nanomover Number, 1 through 16
Description:	Reads the digital input byte for the Nanomotion II motor controller board associated with <mtr>. Each Nanomotion

II motor controller board controls two Nanomovers and has one digital input port (usually used for joystick control). Therefore <mtr> values #1 and #2 will read the same Nanomotion II motor controller board — values #3 and #4 will read the next board - and so on.

The value is returned as a decimal integer corresponding to the sum of the bit values read from the port. The bits have internal pull-up resistors so an unconnected line returns a high. For example, if only the least significant bit in the joystick port is connected to ground, a read of the port would return a "254" (0FE Hex). Only the most significant bit grounded would return "127" (07F Hex). Both grounded would return "126" (07E Hex), etc.

Example: `RDIO,2`

Retrieves the digital input value for Nanomotion controller board associated with Nanomover #2 (addressed to 0300 Hex). If the port is unconnected, the value hardware read value will be 0FF Hex (255 Decimal), the interface card will respond with the ASCII string

`2,255 CRLF`

REBP,<mtr>

Command: Read extended backporch

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Reads the length of the extended backporch in microsteps.

Extended backporches were developed to help settling times in custom(non-Nanomotion II) applications with high-inertia, low-dampening loads like rotary wheels. Most users should not need this functionality and should leave the extended backporch disabled.

Example: `REBP,3`

Retrieves the extended backporch value in microsteps for Nanomover #3. If the extended backporch value is 200, the interface card will respond with the ASCII string

`3,200 CRLF`

REBPE,<mtr>

Command: Read extended backporch enabled

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Reads the whether the extended backporch is enabled or disabled for the specified Nanomover. The final motor position for a move is the same whether the extended backporch is enabled or disabled. The default for Nanomotion II is backporch disabled and most system users should have no reason to change it.

Example: `REBPE,2`

Retrieves the backporch enabled value for Nanomover #2. If the extended backporch is disabled for the Nanomover, the interface card will respond with the ASCII string

`2,OFF CRLF`

RENCC,<mtr>

Command: Read encoder counts

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Reads the encoder counts per motor cardinal steps ratio for the specified Nanomover. The correct ratio must be present in the Nanomover microcontroller for correct functioning of the encoder as a motor stall detection mechanism.

Example: `RENCC,2`

Retrieves the encoder counts per motor step value for Nanomover #2. If the Nanomover microcontroller is correctly set up for a 400 step per revolution Nanomover motor and a 1600 quadrature count per revolution encoder, the interface card will respond with the ASCII string

`2,4 CRLF`

In this case, there would be 4 encoder counts for each motor step.

Note: All the encoder functions in the Nanomotion II libraries require external hardware (an encoder with electrical interface) to work properly.

RENCE,<mtr>

Command: Read encoder enabled

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Reads the whether the encoder checking is enabled or disabled for the specified Nanomover. If the encoder is enabled, the Nanomover microcontroller will check as it moves the Nanomover to make sure that the stepper motor has not stalled. If the motor stalls, the Nanomover is stopped and the encoder error status is set for the Nanomover.

Example: `RENCE,1`

Retrieves the encoder enabled value for Nanomover #1. If encoder stall detection is enabled for Nanomover #1, the interface card will respond with the ASCII string

`1,ON CR(LF)`

Note: All the encoder functions in the Nanomotion II libraries require external hardware (an encoder with electrical interface) to work properly.

RES,<mtr>

Command: Reset motor

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Resets Motor to the default motion profile values. The units will be reset to "MM".

Note: The position is not affected. Neither the physical position of the Nanomover nor the position registers in the microcontroller are changed by this function

This is slightly different from the *RST interface command. This command will change motor settings to bring them back to a normal state. The *RST will only stop the motors without changing any motion settings.

Example: `RES,2`

Resets Nanomover #2 to the default motion parameters (except for position).

RGAIN,<mtr>

Command: Read system gain parameters

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Retrieves the system gain in units per motor revolution for the specified Nanomover. The system gain is defined as the number of units that the mechanical system travels in one revolution of the motor.

A Nanomover moves 0.5 millimeter for each motor revolution. If the units are "MM" (millimeters), the system gain is 0.5 (mm/revolution). If the units are "IN" (inches), the system gain is 0.5/25.4 or about 0.197 (inches/revolution). This function is only needed for custom (non-Nanomotion II) applications. The default value is correct for Nanomovers.

Example: `RGAIN,3`

Retrieves the gain value for Nanomover #3. If the units are millimeters for a standard Nanomover, the interface card will respond with the ASCII string

`3,0.50000 CRLF`

RHCUR,<mtr>

Command: Read holding current

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Returns the holding current for the Nanomover. The holding current will be in the range 0 amps to 2.46 amps. Holding current is the current to the Nanomover motor when the Nanomover is stationary between commanded moves.

See the library function description for more information.

Example: `RHCUR,2`

Retrieves the holding current value for Nanomover #2. If the holding current value is 0.47 amps, the interface card will respond with the ASCII string

`2,0.47000 CRLF`

RLIME,<mtr>

Command: Read limit switches enabled

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Reads whether the hardware limit switches are enabled or disabled for the specified Nanomover. When enabled, the limit switches provide a hardware mechanism for

preventing mechanical "crashes". When the Nanomover microcontroller detects a limit switch, it will decelerate the Nanomover to a stop.

The value is returned as a decimal integer corresponding to the sum of the limit enable values. For example, if only the minimum limit is enabled a limit enable read would return a "1" (01 Hex). Only the index enabled would return "4" (04 Hex). The default condition is minimum and maximum limits enable and the index switch disabled or "3" (03 Hex).

Bit 0:	minimum limit	1 means enabled 0 means disabled
Bit 1:	maximum limit	1 means enabled 0 means disabled
Bit 2:	index switch	1 means enabled 0 means disabled

Example: `RLIME,2`

Retrieves the limit switch enabled values for Nanomover #2. If the minimum and maximum limits are enabled for the Nanomover, the interface card will respond with the ASCII string

`3,3 CRLF`

RLM,<mtr>

Command: Read lost motion compensation

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Read lost motion value. Reads the amount of LMC currently enabled for the specified Nanomover. The value can be in the range of ±10. A lost motion value of zero indicates that LMC is disabled. Any other value indicates the amount and direction of LMC.

Example: `RLM, 1`

The value of lost motion for Nanomover #1 is retrieved. If the LMC value is +2, the interface card will respond with the ASCII string

`1,2 CRLF`

RLT,<mtr>

Command: Read position lock time

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Read position lock time. Reads the length of time to lock the Nanomover rotor into position. The units are in milliseconds.

See the library function description for more information.

Example: `RLM,3`

Retrieves the position lock time value for Nanomover #3. If the lock time value is 100 milliseconds, the interface card will respond with the ASCII string

`3,100 CRLF`

RP,<mtr>

Command: Read position

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Reads the position of the specified Nanomover in the current units.

Example: `RP,2`

Retrieves the position value for Nanomover #2. If the position value is 2.43325 millimeters, the interface card will respond with the ASCII string

`2,2.43325 CRLF`

RRES,<mtr>

Command: read system resolution

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Returns the system resolution for Motor in the currently selected units.

Example: `RRES,1`

Retrieves the system resolution value for Nanomover #1. A standard Nanomover has a default resolution of 10 nm. If the units are millimeters, the interface card will respond with the ASCII string

`1,0.00001 CRLF`

RRINTLK,<mtr>

Command: Release reset interlock

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Resets the reset interlock for the specified Nanomover. The reset interlock is released as part of the boot procedure for the GPIB / RS-232C interface card so the user should not need to use this functionality.

Example: `RRINTLK,2`
Releases the reset interlock for Nanomover #2.

RS,<mtr>

Command: Read status

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Read status of the specified Nanomover. One of the following status values will be returned.

LL	The left (retracted) hardware limit switch is active.
RL	The right (extended) hardware limit switch is active.
IP	The index pulse (bi-directional) hardware limit switch is active.
AL	The Nanomover is at the absolute left (retracted) software stop position.
AR	The Nanomover is at the absolute right (extended) software stop position.
MV	The Nanomover is currently moving.
OK	Normal stationary Nanomover status.
PK	The motor is in the PARKED state. When the Nanomover is parked, it will ignore all commands to move.
EN	A motor stall has been detected with the encoder.
ME	The last move did not finish at its commanded position the Nanomover may have been commanded to stop, or hit a limit switch or a software stop, or whatever).
RS	The Nanomover microcontroller has been reset. This is an error condition possibly resulting from a power supply "brown out".

Example: `RS,2`

Retrieves the status for Nanomover #2. If the Nanomover is moving, the interface card will respond with the ASCII string

`2,MV CRLF`

RSCUR,<mtr>

Command: Read stepping current

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Returns the stepping current for the Nanomover. The stepping current will be in the range 0-2.46 A. Stepping current is the current to the Nanomover motor when the Nanomover is moving.

See the library function description for more information.

Example: `RSCUR,3`

Retrieves the stepping current value for Nanomover #3. If the stepping current value is 0.90 amps, the interface card will respond with the ASCII string

`3,0.90000 CRLF`

RSTENC,<mtr>

Command: Reset encoder

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Resets the encoder interface for the specified Nanomover. After a motor stall is detected by the Nanomover microcontroller, this command will re-synchronize the Nanomover motor and encoder interface and re-enable Nanomover movement.

See the library function description for more information.

Example: `RSTENC,1`

Resets the encoder interface for Nanomover #1. If a motor stall had been detected, this command will re-enable motion for that Nanomover.

Note: All the encoder functions in the Nanomotion II libraries require external hardware (an encoder with electrical interface) to work properly.

RSTP,<mtr>

Command: Read motor steps

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Retrieves the number of cardinal motor steps per revolution for the specified Nanomover. Normal Nanomovers use 400 steps per revolution motors.

Example: `RSTP,2`

Retrieves the motor steps per revolution value for Nanomover #2. For a normal Nanomover, the value is 400 steps per revolution so the interface card will respond with the ASCII string

`2,400 CRLF`

RU,<mtr>

Command: Read units

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Reads the type of units in use. The returned information will be one of the following strings: NM for nanometers, MI for microns, MM for millimeters, CM for centimeters, UI for micro-inches, ML for mils, or I for inches.

Example: `RU,4`

Retrieves the units base for Nanomover #4. If the units are millimeters, the interface card will respond with the ASCII string

`4,MM CRLF`

RUSTP,<mtr>

Command: Read microsteps

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Retrieves the microsteps per cardinal motor step value for the current Nanomover.

Example: `RUSTP,2`

Retrieves the microsteps per cardinal motor step value for Nanomover #2. With the default Nanomotion value of 125 microsteps per motor step so the interface card will respond with the ASCII string

`2,125 CRLF`

RV<ramp>,<mtr>

Command: Read velocity

Parameters: <ramp> — Velocity/Acceleration Ramp Segment Number, 1 through 4

<mtr> — Nanomover Number, 1 through 16

Description: Reads the velocity value in the current units (per second) for the specified velocity/acceleration ramp segment for the specified Nanomover. The <ramp> parameter specifies ramp segment 1, 2, 3, or 4.

Example: `RV1,2`

Retrieves the velocity value for velocity/acceleration ramp #1 for Nanomover #2. If the velocity value is 2.5 mm/s, the interface card will respond with the ASCII string

`2,2.50000 CRLF`

RVER,<mtr>

Command: Read EPROM version

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Will return the version of the microcontroller EPROM for the specified Nanomover.

Example: `RVER,2`

Retrieves the EPROM version for Nanomover #2. If the EPROM is version 3.01, the interface card will respond with the ASCII string

`1,3.01000 CRLF`

S -or- S<mtr>

Command: Stop

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: This command stops all motion. The command S will stop all moving Nanomovers. Sometimes it is convenient to stop a single Nanomover. The command S<mtr> will stop only the specified Nanomover.

Example: `S1`

This command will stop Nanomover #1 but leave all others unaffected.

UA -or- U,<mtr>

Command: Unpark

Parameters: <mtr> — Nanomover Number, 1 through 16

Description: Unparks the specified Nanomovers. UA will unpark all parked Nanomovers. U,<mtr> will unpark only the specified Nanomover. If the specified motor(s) are parked, this command will unpark the motor(s) using the position and park offsets stored in the EEPROM on the interface card. If the requested motor is already unparked or has not been parked, this command will be ignored.

Example: `U,2`

Unparks Nanomover #2.The current position and the offset necessary to unpark the Nanomover will be restored from the interface card.

`UA`

Unparks all Nanomovers. Again, the information necessary to unpark each Nanomover is restored from the interface card.

WA<ramp>,<mtr>,<accel>

Command: Write acceleration

Parameters: <ramp> — Velocity/Acceleration Ramp Segment Number, 1 through 4

<mtr> — Nanomover Number, 1 through 16

<accel> — Acceleration Value

Description: Write the acceleration to the specified velocity/acceleration ramp segment. The <ramp> parameter specifies ramp segment 1, 2, 3, or 4. The acceleration value must be within the range specified for the Nanomotion II system, otherwise the value will be limited to be within the allowable range.

Example: `WA1,1,100.0`

This command will set the acceleration for ramp #1 on Nanomover #1 to 100.0 units per second per second. If the units are millimeters, ramp #1 for Nanomover #1 will be set to 100.0 mm per second per second.

WAL,<mtr>,<absleft>

Command: Write absolute left

Parameters: <mtr> — Nanomover Number, 1 through 16

<absleft> — Absolute Left Stop Position

Description: This instruction sets the position of the absolute left stop for the designated Nanomover. Any time a Nanomover is commanded to a position less than the <absleft> value, the motor will stop at the absolute left stop position rather than moving to the commanded position..

Example: `WAL,3,−1.0`

The absolute left stop position of Nanomover #3 will be set to −1.0 units.

WAR,<mtr>,<absright>

Command: Write absolute right

Parameters: <mtr> — Nanomover Number, 1 through 16

<absright> — Absolute Right Stop Position

Description: This instruction sets the position of the absolute right stop for the designated Nanomover. Any time a Nanomover is commanded to a position greater than the <absright> value, the motor will stop at the absolute right stop position rather than moving to the commanded position.

Example: `WAR,1,15.0`

The absolute right stop position of Nanomover #1 will be set to 15.0 units.

WB,<mtr>,<bvel>

Command: Write base velocity

Parameters: <mtr> — Nanomover Number, 1 through 16

<bvel> — Base Velocity Value

Description: This command sets the base velocity for the specified motor to the value <bvel>. The base velocity should be within the range specified for the Nanomotion II system, otherwise the value will be limited to be within the allowable range.

Example: `WB,1,0.100`

This command will set the base velocity for Nanomover #1 to 0.100 units per second. If the current units are millimeters, the base velocity is set to 0.100 mm per second.

WBPE,<mtr>,<on/off>

Command: Enable backporch

Parameters: <mtr> — Nanomover Number, 1 through 16

<on/off> — Enable string, "ON" or "OFF"

Description: Sets the whether the backporch function is enabled (ON) or disabled (OFF) (the default status is enabled) for the specified Nanomover. The final motor position for a move is the same whether the backporch is enabled or disabled. The backporch is a short move at the base velocity at the end of each commanded move and is a normal feature of Nanomotion II — most system users should have no reason to change it.

Example: `WBPE,2,ON`

Enables the backporch functionality for Nanomover #2.

WCURE,<mtr>,<on/off>

Command: Enable motor current

Parameters: <mtr> — Nanomover Number, 1 through 16

<on/off> — Enable string, "ON" or "OFF"

Description: Enables (ON) or Disables (OFF) the motor current for the specified Nanomover (the default status is enabled). The current is enabled as part of the GPIB / RS232-C interface board boot process. Most users should have no reason to turn the motor current on or off — the holding current functionality will reduce the current while the motor is stationary to minimize thermal problems.

Example: `WCURE,1,OFF`

Will turn the current off for Nanomover #1.

WEBP,<mtr>,<ext. bporch>

Command: Write extended backporch

Parameters: <mtr> — Nanomover Number, 1 through 16

	<ext. bporch> — Extended backporch value
Description:	Sets the length of the extended backporch value for the specified Nanomover in microsteps.
	Extended backporches were developed to help settling times in custom (non-Nanomotion II) applications with high-inertia, low-dampening loads like rotary wheels. Most users should not need this functionality and should leave the extended backporch disabled.
Example:	`WEBP,4,100`
	Sets the extended backporch to 100 microsteps for Nanomover #4.

WEBPE,<mtr>,<on/off>

Command:	Enable extended backporch
Parameters:	<mtr> — Nanomover Number, 1 through 16
	<on/off> — Enable string, 'ON' or 'OFF'
Description:	Sets the whether the extended backporch function is enabled (ON) or disabled (OFF) for the specified Nanomover (the default status is disabled).
	Extended backporches were developed to help settling times in custom(non-Nanomotion II) applications with high-inertia, low-dampening loads like rotary wheels. Most users should not need this functionality and should leave the extended backporch disabled.
Example:	`WEBPE,2,ON`
	Enables the backporch functionality for Nanomover #2.

WENCC,<mtr>,<encoder cnt>

Command:	Write encoder counts
Parameters:	<mtr> -— Nanomover Number, 1 through 16
	<encoder cnt> — Encoder counts per cardinal motor step value
Description:	Sets the encoder counts per motor cardinal steps ratio for the specified Nanomover.The correct ratio must be present in the Nanomover microcontroller for correct functioning of the encoder as a motor stall detection mechanism. Supported ratios are: 2, 4, 8, and 16.

For most users, it may be easier to use the DENCC or DSTP functions to let the Nanomover microcontroller determine the ratio for itself.

Example: `WENCC,2,4`

Sets the encoder counts per motor step value for Nanomover #2. If the Nanomover microcontroller has a 400 step per revolution Nanomover motor and a 1600 quadrature count per revolution encoder, this value would correctly set up the encoder functionality with 4 encoder counts for each motor step.

Note: All the encoder functions in the Nanomotion II libraries require external hardware (an encoder with electrical interface) to work properly.

WENCE,<mtr>,<on/off>

Command: Enable encoder

Parameters: <mtr> — Nanomover Number, 1 through 16

<on/off> — Enable string, "ON" or "OFF"

Description: Sets the whether the encoder stall detection mechanism is enabled (ON) or disabled (OFF) for the specified Nanomover (the default status is disabled).

See the library function description for more information.

Example: `WENCE,2,ON`

Enables the encoder checking functionality for Nanomover #2.

Note: All the encoder functions in the Nanomotion II libraries require external hardware (an encoder with electrical interface) to work properly.

WGAIN,<mtr>,<gain value>

Command: Write system gain

Parameters: <mtr> — Nanomover Number, 1 through 16

<gain value> — System gain value

Description: Sets the system gain in units per motor revolution for the specified Nanomover. The system gain is defined as the number of units that the mechanical system travels in one revolution of the motor.

A Nanomover moves 0.5 millimeter for each motor revolution. If the units are "MM" (millimeters), the system

gain should be 0.5 (mm / revolution). If the units are "IN" (inches), the system gain is 0.5/25.4 or about 0.197 (inches/revolution).This function is only needed for custom (non-Nanomotion II) applications.The default value is correct for Nanomovers.

Example: `WGAIN,3,0.5`

Sets the gain value for Nanomover #3. If the units are millimeters for a standard Nanomover, the system gain should be 0.5.

WHCUR,<mtr>,<current>

Command: Write holding current

Parameters: <mtr> — Nanomover Number, 1 through 16

<current> — Holding current for the Nanomover

Description: Sets the holding current for the specified Nanomover. The holding current must be in the range 0 amps to 2.46 amps. Holding current is the current to the Nanomover motor when the Nanomover is stopped between moves. The default is correct for Nanomovers and most users should have no reason to change it

Example: `WHCUR,3,0.47`

Sets the holding current for Nanomover #3 to 0.47 amps.

WLIME,<mtr>,<on/off bits>

Command: Enable limit switches

Parameters: <mtr> — Nanomover Number, 1 through 16

<on/off bits> — Enable / Disable bits for the hardware limit switches

Description: Sets whether the limit switch inputs are enabled or disabled for the specified Nanomover. When enabled, the limit switches provide a hardware mechanism for preventing mechanical "crashes". When the Nanomover microcontroller detects a limit switch, it will decelerate the Nanomover to a stop.

The <enable bits> is a decimal integer corresponding to the sum of the limit enable values. For example, if only the

minimum limit should be enabled, a limit enable value of "1" (01 Hex) would be used. Only the index enabled is "4" (04 Hex). The default condition is minimum and maximum limits enable and the index switch disabled or "3" (03 Hex).

Bit 0:	minimum limit	1 means enabled 0 means disabled
Bit 1:	maximum limit	1 means enabled 0 means disabled
Bit 2:	index switch	1 means enabled 0 means disabled

Example: `WLIME,2,3`

Enables the minimum and maximum limit switches Nanomover #2.

WLM,<mtr>,<lmc>

Command: Write lost motion compensation

Parameters: <mtr> — Nanomover Number, 1 through 16

<lmc> — LMC value, -10 through $+10$

Description: This commands sets the lost motion compensation (LMC) value to be used for the specified Nanomover. The LMC value should be an integer in the range of ± 10; otherwise the value will be limited to be within the allowable range. A value of 0 will disable LMC. Any other value will enable it. Each unit of LMC corresponds to 10µm of Nanomover travel. The sign of the LMC value specifies the direction of the LMC at the end of each move.

Example: `WLM,1,2`

The lost motion compensation will be set to $+2$ ($+20$ µm) for Nanomover #1.

WLT,<mtr>,<lock>

Command: Write lock

Parameters: <mtr> — Nanomover Number, 1 through 16

<lock> — Position Lock Time value

Description: This command sets the duration of time to lock the Nanomover into a new position. The units are always in milliseconds. The lock time value should be within the

range of 0 to 250 milliseconds. If the lock time exceeds this range, the position lock time will be limited to be within the allowable range.

Example: `WLT,1,100`

The lock time will be set to 100 ms for Nanomover #1.

WP,<mtr>,<pos>

Command: Write position

Parameters: <mtr> — Nanomover Number, 1 through 16

<pos> — Position Value

Description: This command sets the current position for the specified Nanomover to the <pos> value. The Nanomover does not move, only the software position variable is modified. The position should be within the range specified for the Nanomotion II system, otherwise the value will be limited to be within the allowable range.

Example: `WP,1,10.0`

This command will set the position for Nanomover #1 to 10.0 units. If the current units are millimeters, the software position will be set to 10.0 mm — the Nanomover will not move.

WSCUR,<mtr>,<current>

Command: Write stepping current

Parameters: <mtr> — Nanomover Number, 1 through 16

<current> — Holding current for the Nanomover

Description: Sets the stepping current for the specified Nanomover. The stepping current must be in the range 0 amps to 2.46 amps. Stepping current is the current to the Nanomover motor when the Nanomover is moving. The default value is correct for Nanomovers and most users should have no reason to change it.

Example: `WSCUR,3,0.90`

Sets the stepping current for Nanomover #3 to 0.90 A

WSTP,<mtr>,<steps>

Command: Write motor steps

Parameters: <mtr> — Nanomover Number, 1 through 16

<steps> — Number of motor steps for the Nanomover motor

Description: Sets the number of cardinal motor steps per revolution for the specified Nanomover. Normal Nanomovers use 400 steps per revolution motors and most users should not have any reason to change this value.

Example: `WSTP,2,400`

Sets the motor steps per revolution value for Nanomover #2 to 400 steps per revolution.

WU,<mtr>,<units>

Command: Write units

Parameters: <mtr> — Nanomover Number, 1 through 16

<units> — Units string

Description: This command defines the units for a particular Nanomover. <units> must be one of the following strings: NM for nanometers, MI for microns, MM for millimeters, CM for centimeters, UI for microinches, ML for mils, or I for inches. All other input or output motion control values (position, velocity, etc.) for the specified Nanomover will be interpreted in terms of the specified units.

See the library function description for more information.

Example: `WU,2,MM`

Sets the units base for Nanomover #2 to millimeters.

WUSTP,<mtr>,<µsteps>

Command: Write microsteps

Parameters: <mtr> — Nanomover Number, 1 through 16

<µsteps> — Microsteps per cardinal motor step value

Description: Sets the microsteps per cardinal motor step for the specified Nanomover microcontroller. The microstep per step value must be one of the supported values: 25, 50, 100, 125, or 250. The default value is 125 and most users should

have no reason to change it.

See the library function description for more information.

Note: When the microstep resolution is changed, the Nanomover software position will be reset to zero (but the Nanomover will not physically move).

Example: `WUSTP,2,125`

Sets the microsteps per cardinal motor step value for Nanomover #2 to 125.

WV<ramp>,<mtr>,<vel>

Command: Write velocity

Parameters: <ramp> — Velocity/Acceleration Ramp Segment Number, 1 through 4

<mtr> — Nanomover Number, 1 through 16

<vel> — Velocity Value

Description: Write the velocity to the specified velocity/acceleration ramp segment. The <ramp> parameter specifies ramp segment 1, 2, 3, or 4. The velocity value must be within the range specified for the Nanomotion II system, otherwise the value will be limited to be within the allowable range.

Example: `WV1,1,2.5`

This command will set the velocity for acceleration ramp #1 on Nanomover #1 to 2.5 units per second. If the units are millimeters, the ramp #1 for Nanomover #1 will be set to 2.5 mm per second.

6.2 Programming with the PC-Link Interface

6.2.1 Introduction

For some applications, the power and flexibility of the Nanomotion II Control Program may not be appropriate. Instead, users may prefer to control the Nanomotion II system using a custom application. Recognizing this need, several programming options have been made available. First, users can use a comprehensive DLL (dynamic-link library) to construct Windows-based applications in nearly any Windows development environment. Second, an MS-DOS based application can be created using C and a linkable library of Nanomotion II functions.

6.3 Dynamic-Link Library (DLL)

For users who want to control Nanomotion II using a custom Windows application, whether because it is part of a larger system of instrumentation or simply to provide unique functionality, a DLL (dynamic-link library) containing a comprehensive command library has been provided. Any compiled language, such as Visual Basis or C/C++, that can access a Windows DLL should be able to use the library. You must consult the documentation for your language to determine how to use the Nanomotion II DLL.

6.4 MS C7.0 Library for DOS Applications

6.4.1 Introduction

The Nanomotion II libraries are sets of "C" language functions that allow the user to control the Nanomotion II position system using the Microsoft "C" and "C++" languages — either in a standalone MS-DOS .EXE program or a Windows application using the DLL.

This documentation assumes that the user is familiar with the operation of the Nanomotion II system and its associated terminology. The Nanomotion II manual details the configuration of the Nanomotion II system and the definition of parameters.

All Library functions begin with "nl_" that stands for "Nanomotion II library". The "nl_" prefix will help distinguish the Nanomotion II library functions in your source code. The rest of the function name is descriptive of the function's use. For example, nl_read_position reads the position of a Nanomover. This should make using the libraries less of an exercise in memory, and produce final code that is easy to understand and maintain. Of course, the user is free to redefine function names to those that are most descriptive for their application.

The libraries are compiled with Microsoft "C", using the most current version available at the time. The compiler version is listed in an ASCII text file called READ.ME on the release disk. A complete set of MS-DOS libraries is provided which conforms to the Microsoft defined models. The standard libraries have the naming convention "XMSCNANO.LIB" where "X" indicates the memory model.

The supplied libraries are:

SMSCNANO.LIB	Small model library
MSCNANO.LIB	Medium model library
CMSCNANO.LIB	Compact model library
LMSCNANO.LIB	Large model library
HMSCNANO.LIB	Huge model library

There is one DLL library and a static import library. The DLL must be located in your Windows directory or in the directory with your executable program at run time. The import library can be linked with your executable program to make accesses to the DLL easier to program (refer to the Microsoft compiler documentation for more information on using dynamic link libraries).

There is also one header file, NANOLIB.H, which should be included whenever the Nanomotion II libraries are used. This header file contains Microsoft compatible prototyping statements for all the Nanomotion II library commands. It also contains #define statements for commonly used Nanomotion II constants.

Demonstration Program

The release diskette(s) contain a demonstration program that uses the Nanomotion II "C" library.

NANODEMO.C	"C" Source File

This file contains programming examples and comments regarding the use of the Nanomotion II libraries. Examination of the example program is probably the quickest way to become familiar with the use of the Nanomotion II libraries. Use them as a guide in writing your own programs.

NANODEMO.MAK	Visual C++ make file used to compile and link the demo

6.4.2 Functions by Type

Move Parameters:

nl_get_absleft	nl_read_absright	nl_write_accel
nl_read_absleft	nl_write_absright	nl_get_bvel
nl_write_absleft	nl_read_accel	nl_read_bvel
nl_get_absright	nl_get_accel	nl_write_bvel

Miscellaneous:

nl_reset	nl_setbase
nl_get_motor_address	nl_initialized

Limit Switches:

nl_enable_limit_switches	nl_read_lock	nl_read_units
nl_get_limit_switches_enabled	nl_write_lock	nl_write_units
nl_read_limit_switches_enabled	nl_get_position	nl_get_vel
nl_get_lmc	nl_read_position	nl_read_vel
nl_read_lmc	nl_write_position	nl_write_vel
nl_write_lmc	nl_get_status	
nl_get_lock	nl_read_status	

Digital I/O:

nl_get_joysticks	nl_get_nano_ii_dio
nl_read_joysticks	nl_read_nano_ii_dio

Motor Commands:

nl_move_absolute	nl_stop	nl_unpark
nl_move_relative	nl_park	

Initialization

nl_init	nl_save_cfg	nl_get_eprom_version
nl_exit	nl_get_base_address	nl_read_eprom_version
nl_get_initialized	nl_read_base_address	nl_get_microsteps
nl_read_initialized	nl_write_base_address	nl_read_microsteps
nl_write_initialized	nl_get_dll_version	nl_write_microsteps
nl_restore_cfg	nl_read_dll_version	

nl_enable_limit_switches <mtr>, <min><max>

Command: Enable/disable limit switches

Parameters: <mtr> — Nanomover number, 1 through 16

<min> — Minimum limit switch

0 for disabled

1 for enabled

<max> — Maximum limit switch

0 for disabled

1 for enabled

Description: Enables or disables the limit switches for the specified motor. The Nanomotion II system does not need limit switches, the Absolute Left and Absolute Right stops will control the motion range of the Nanomover.The default for Nanomotion II is limit switches enabled. The switches are defined in the Nanomotion II system as normally-open so if no switch hardware is connected, the microcontroller will allow motion. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_enable_limit_switches 4,11`

Returns 0 if operation is successful

nl_exit

Command: Exit Nanomotion library routine.

Parameters: None

Description: This function must be called when your program is through using the Nanomotion II library routines and before your program exits. This function resets various interrupts set by **nl_init**. If these interrupts are not reset when the program exits, the computer may “hang” or programs executed later may not function properly.

Example: `nl_exit`

nl_get_absleft <mtr>

Command: Get absolute left stop position

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the absolute left stop position for the Nanomover. Maps internally in the library to **nl_read_absleft** but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_get_absleft 4`

Returns the absolute left stop position for Nanomover 4. If there is an error, the function returns − 1.0

nl_get_absright

Command: Get absolute right stop position

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the absolute right stop position for the Nanomover. Maps internally in the library to nl_read_absright but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example:
```
nl_get_absright 5
```
Returns the absolute right stop position for Nanomover 5. If there is an error, the function returns -1.0

nl_get_accel <mtr>, <ramp>

Command: Get acceleration value

Parameters: <mtr> — Nanomover number, 1 through 16

<ramp> — Acceleration/Velocity Ramp segment, 1 through 4

Description: Returns the acceleration value for the specified ramp for the Nanomover. Maps internally in the library to **nl_read_accel** but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example:
```
nl_get_accel 12,3
```
Returns the acceleration value for the third segment of the twelfth motor. If there is an error, the function returns -1.0

nl_get_base_address

Command: Get current base address for the system

Parameters: None

Description: Returns the current base address for the system. Maps internally in the library to **nl_read_base_address** but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_get_base_address`

Returns the current system base address value. If there is an error, the function returns 0xFFFF

nl_get_bvel <mtr>

Command: Get base velocity

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the base velocity for the Nanomover. Maps internally in the library to **nl_read_bvel** but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_get_bvel 6`

Returns the base velocity for the Nanomover. If there is an error, the function returns − 1.0

nl_get_dll_version

Command: Get library version

Parameters: None

Description: Returns the library version as an integer. Maps internally in the library to **nl_read_dll_version** but does not require the user to supply a pointer to a variable buffer. The same function name is used for both the DLL and "C" library version numbers. The version is expressed as an integer number(word). That is, the upper byte of the version word is the most significant digit, the lower byte is the least significant digit. For example, version number 0x0300 would be interpreted as 3.0, 0x0301 would be 3.1, etc.

Example: `nl_get_dll_version`

Returns the library version as an integer. If there is an error, the function returns − 1

nl_get_eprom_version <mtr>

Command: Get eprom version

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the Z8 EPROM version for the specified Nanomover. Maps internally in the library to **nl_read_eprom_version** but does not require the user to

supply a pointer to a variable buffer. The version is expressed as an integer number (word). That is, the upper byte of the version word is the most significant digit, the lower byte is the least significant digit. For example, version number 0x0300 would be interpreted as 3.0, 0x0301 would be 3.1, etc. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_get_eprom_version 3`

Returns an integer firmware version for the Z8 microcontroller. If there is an error, the function returns − 1

nl_get_initialized

Command: Get initialization call status

Parameters; None

Description: Returns the flag whether nl_init has been called or not (so programs at runtime can know when not to call it again). Maps internally in the library to **nl_read_initialized** but does not require the user to supply a pointer to a variable buffer.

Example: `nl_get_initialized`

Returns 1 if nl_init has been already called. Returns 0 if nl_init has not been called. If there is an error, the function returns − 1

nl_get_joysticks <mtr>

Command: Get joystick value

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the input from the Nanomotion I joystick port for the specified Nanomover. Maps internally in the library to **nl_read_joysticks** but does not require the user to supply a pointer to a variable buffer. This function only works on Nanomotion I hardware. Nanomotion II applications should use the nl_read_nano_ii_dio function. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_get_joysticks 2`

Returns the current joystick value for motor 2. If there is an error, the function returns 0xFF

Note: There is only one joystick port shared by motors 1 & 2, etc. Calling this function with either of the motors for a specific board will return the data from the same dio port.

nl_get_limit_switches_enabled <mtr>

Command: Get limit switch enabled status

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the limit switch enabled bits for the specified Nanomover. Maps internally in the library to **nl_read_limit_switches_enabled** but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_get_limit_switches_enabled 2`

Returns 00 if neither limit is enabled

Returns 10 if only minimum limit is enabled

Returns 01 if only maximum limit is enabled

Returns 11 if both limits are enabled

Returns −1 if there is an error

nl_get_lmc <mtr>

Command: Get lost motion compensation value

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the lost motion compensation value for the specified Nanomover. Maps internally in the library to **nl_read_lmc** but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_get_lmc 6`

Returns the LMC value for motor 6. If there is an error, the function returns −1

nl_get_lock <mtr>

Command: Get position lock time

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the position lock time for the specified

Nanomover. Maps internally in the library to nl_read_lock but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_get_lock 5`

Returns the position lock time for the Nanomover in milliseconds. If there is an error, the function returns − 1

nl_get_microsteps <mtr>

Command: Get number of microsteps per step

PARAMETERS: <MTR> — NANOMOVER NUMBER, 1 THROUGH 16

Description: Returns the microsteps per step for the specified Nanomover. Maps internally in the library to **nl_read_microsteps** but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_get_microsteps 1`

Returns the microsteps per step for the motor. If there is an error, the function returns − 1

nl_get_motor_address <mtr>

Command: Get I/O address

PARAMETERS: <MTR> — NANOMOVER NUMBER, 1 THROUGH 16

Description: Returns the PC I/O address of the specified Nanomover. Each Nanomover occupies four(4) locations in the PC's I/O space. This function will return the lowest of those locations. The Nanomovers are always accessed by their motor numbers and this function is only used as a convenience to help the user identify possible addressing conflicts in his PC system.

Example: `nl_get_motor_address 16`

Returns an integer address of the Nanomotion II board. If there is an error, the function returns − 1

nl_get_motor_steps <mtr>

Command: Get cardinal steps per revolution

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the cardinal steps per revolution for the specified motor. Maps internally in the library to **nl_read_motor_steps** but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_get_motor_steps 3`

Returns the steps per revolution for the motor. If there is an error, the function returns − 1

nl_get_nano_ii_dio <mtr>

Command: Get DIO input

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the input from the Nanomotion II DIO port for the specified Nanomover. Maps internally in the library to **nl_read_nano_ii_dio** but does not require the user to supply a pointer to a variable buffer. This function only works on Nanomotion II hardware. Nanomotion I applications should use the nl_read_joysticks function. If the motor is currently moving, this function will wait for it to stop before proceeding.

Note: There is only one joystick port shared by motors 1 & 2, etc. Calling this function with either of the motors for a specific board will return the data from the same dio port.

Example: `nl_get_nano_ii_dio 4`

Returns the dio value from motor 4. If there is an error, the function returns 0xFF

nl_get_position <mtr>

Command: Get current position

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the current position for the specified motor. Maps internally in the library to **nl_read_position** but does not require the user to supply a pointer to a variable buffer.

Example: `nl_get_position 4`

Returns the current position of the motor 4. If there is an error, the function returns − 1.0

nl_get_status <mtr>

Command: Get motor status.

Parameters: <mtr> — Nanomover number, 1 through 16

Description: Returns the current status word for the specified motor. Maps internally in the library to **nl_read_status** but does not require the user to supply a pointer to a variable buffer. See the nl_read_status function for descriptions of the status word.

Example:
```
nl_get_status 3
```
Returns the current status for the Nanomover. If there is an error, the function returns 0xFF

nl_get_vel <mtr>,<ramp>

Command: Get ramp velocity

Parameters: <mtr> — Nanomover number, 1 through 16

<ramp> - Acceleration/velocity ramp segment 1 through 4

Description: Returns the velocity for the specified ramp for the specified motor. Maps internally in the library to **nl_read_velocity** but does not require the user to supply a pointer to a variable buffer. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example:
```
nl_get_vel 11, 2
```
Returns the velocity value of the second ramp for motor 11. If there is an error, the function returns −1.0

nl_init

Command:

Parameters: None

Description: Initializes all arrays and variables used internally by the Nanomotion II libraries. This should be called near the beginning of your program, as all other Nanomotion II functions assume correct variables. It MUST be called prior to using any of the routines or moving any motors, and MUST be called only once before calling nl_exit. In addition to initializing software variables, nl_init actually performs a hardware reset on the Nanomotion II card. This guarantees that the hardware is also initialized properly whenever this

function is called. If nl_init returns an error condition (non-0 value), nl_exit must still be called before exiting your program since the interrupts may have been changed.

Example: `nl_init`

Returns 0 if operation successful; non-0 if operation unsuccessful or no cards present.

nl_move_absolute <mtr>,<dest>

Command: Move to specified destination

Parameters: <mtr> — Nanomover number, 1 through 16

<dest> — Destination of the move

Description: Moves the motor to the specified destination. The variable dDestination is in terms of currently selected distance units (such as mm or nm). The + and − directions of motion are as specified for the Nanomovers (+ is extension of the micrometer shaft and - is retraction of the shaft). The move will be performed using the defined motion profile using the velocities, accelerations, and position limits in the Z8 microcontroller. This function will not move a parked motor. If the motor is currently moving, this function will wait for it to stop before initiating a new movement.

Example: `nl_move_absolute 4, −3`

Returns 0 if motor has started, non-0 if motor is parked or an error occurs.

nl_move_relative<mtr>,<dist>

Command: Move a specified distance

Parameters: <mtr> — Nanomover number, 1 through 16

<dist> — Destination of the move

Description: Moves the Nanomover the specified distance. The variable dDistance is in terms of currently selected units(such as mm or nm). The + and − directions of motion are as specified for the Nanomovers (+ is extension of the micrometer shaft and − is retraction of the shaft). The move will be performed using the defined motion profile using the velocities, accelerations, and position limits in the Z8 microcontroller. This function will not move a parked motor. If the motor is currently moving, this function will wait for it to stop before initiating a new movement.

Example: `nl_move_relative 12,6`

Returns 0 if motor has started, non-0 if motor is parked or an error occusr.

nl_park <mtr>

Command:

Parameters: <mtr> — Nanomover number, 1 through 16; −1 is all motors

Description: If nMotor is 1 through 16, this function parks that motor. If nMotor is −1, this function will park all motors. If the motor to be parked is currently moving, this function will wait for it to stop before proceeding. The function nl_park moves the motor to it's closest stable(detent) position in the −(retracting) direction. All park information is stored in internal variables and is not written to disk by this function. Use nl_save_cfg to save the initialization.

Example: `nl_park 6`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_read_absleft <mtr>,<dpLeftLimit>

Command: Reads absolute left stop position

Parameters: <mtr> — Nanomover number, 1 through 16

dpLeftLimit> - Pointer to a double variable for data return

Description: Reads the absolute left limit of travel in current units for a specified motor. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_read_absleft 4,dpLeftLimit`

Returns the current absolute left stop value of motor 4 in the location pointed to by **dpLeftLimit**.

nl_read_absright <mtr>,<dpRightLimit>

Command: Read absolute right stop position in the location pointed to by dpLeftLimit.

Parameters: <mtr> — Nanomover number, 1 through 16

<dpRightLimit> — pointer to a double variable for data

	return
Description:	Reads the absolute right limit of travel in current units for a specified motor. If the motor is currently moving, this function will wait for it to stop before proceeding.
Example:	`nl_get_absright 5`
	Returns the current absolute right limit of travel for motor 5 in the location pointed to by dpRightLimit.

nl_read_accel <mtr>, <ramp>,<dpAcceleration>

Command:	Read acceleration value
Parameters:	<mtr> — Nanomover number, 1 through 16
	<ramp> — Acceleration/Velocity Ramp segment, 1 through 4
	<dpAcceleration> — Pointer to a double variable for data return
Description:	Reads the acceleration value for the specified ramp for the Nanomover. If the motor is currently moving, this function will wait for it to stop before proceeding.
Example:	`nl_read_accel 12,3,dpAcceleration`
	Returns the acceleration value for the third segment of the twelfth motor in the location pointed to by **dpAcceleration**.

nl_read_base_address <upBaseAddress>

Command:	Read current base address for the system
Parameters:	<upBaseAddress> — Pointer to unsigned variable for the data return
Description:	Reads the current base I/O address for the system. If the motor is currently moving, this function will wait for it to stop before proceeding.
Example:	`nl_read_base_address upBaseAddress`
	Returns the current system base address value in the location pointed to by upBaseAddress.

nl_read_bvel <mtr>,<dpBaseVelocity>

Command:	Read base velocity
Parameters:	<mtr> — Nanomover number, 1 through 16

<dpBaseVelocity> — Pointer to a double variable for the data return

Description: Reads the base velocity for the Nanomover. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_read_bvel 6, dpBaseVelocity`

Returns the current base velocity setting in the location pointed to by **dpBaseVelocity**.

nl_read_dll_version <npVersion>

Command: Read library version

Parameters: <npVersion> -— Pointer to an integer variable for the data return

Description: Returns the library version as an integer. The same function name is used for both the DLL and "C" library version numbers. The version is expressed as an integer number(word). That is, the upper byte of the version word is the most significant digit, the lower byte is the least significant digit. For example, version number 0x0300 would be interpreted as 3.0, 0x0301 would be 3.1, etc.

Example: `nl_read_dll_version npVersion`

Returns the library version in the location pointed to by **npVersion**.

nl_read_eprom_version <mtr>,<npVersion>

Command: Read eprom version

Parameters: <mtr> — Nanomover number, 1 through 16

<npVersion> - Pointer to an integer variable for the data return

Description: Reads the Z8 EPROM version for the specified motor. The version is expressed as an integer number(word). That is, the upper byte of the version word is the most significant digit, the lower byte is the least significant digit. For example, version number 0x0300 would be interpreted as 3.0, 0x0301 would be 3.1, etc. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_read_eprom_version 3,npVersion`

Returns the EPROM version for the motor in the location pointed to by **npVersion**.

nl_read_initialized <npInitialized>

Command: Read initialization call status

Parameters; data return: <npInitialized> — Pointer to an integer variable for the

Description: Reads the flag whether nl_init has been called or not (so programs at runtime can know when not to call it again).

Example:
```
nl_read_initialized npInitialized
```
Returns the initialized flag value in the location pointed to by npInitialized. It returns 1 if nl_init has been already called. Returns 0 if nl_init has not been called.

nl_read_joysticks <mtr>,<upJoyvalue>

Command: Read joystick value

Parameters: <mtr> — Nanomover number, 1 through 16

<upJoyvalue> - Pointer to the location for the data return

Description: Reads the input from the Nanomotion I joystick port for the specified motor. If the motor is currently moving, this function will wait for it to stop before proceeding. This function only works on Nanomotion I hardware. Nanomotion II applications should use the nl_read_ii_dio function.

Example:
```
nl_read_joysticks 2,upJoyvalue
```
Returns the current joystick value for motor 2 in the location pointed to by **upJoyvalue**.

Note: There is only one joystick port shared by motors 1 & 2, etc. Calling this function with either of the motors for a specific board will return the data from the same dio port.

nl_read_limit_switches_enabled <mtr>,<npEnabled>

Command: Read limit switch status

Parameters: <mtr> — Nanomover number, 1 through 16

<npEnabled> — Pointer to an integer variable for the data return

Description: Reads whether or not the limit switches are enabled. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_read_limit_switches_enabled 1, npEnabled`

Returns the limits active value in the location pointed to by `npEnabled`.

Returns 00 if neither limit is enabled

Returns 10 if only minimum limit is enabled

Returns 01 if only maximum limit is enabled

Returns 11 if both limits are enabled

nl_read_lmc <mtr>,<npLmcValue>

Command: Read lost motion compensation value

Parameters: <mtr> — Nanomover number, 1 through 16

<npLmcValue> — Pointer to an integer variable for the data return

Description: Reads lost motion compensation for a Nanomover in units of 10 microns (range from -10 to $+10$). LMC does not changed if units are changed, it always in terms of 10 micron units. For example, an LMC value of $+10$ means that lost motion compensation is in the $+$ direction, and it is $10 \times 10\mu m$ or 100 μm in length. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_read_lmc 4,npLmcValue`

Returns the current lost motion value in the location pointed to by **npLmcValue**

nl_read_lock <mtr>,<npLockTime>

Command: Read position lock time

Parameters: <mtr> — Nanomover number, 1 through 16

<npLockTime> — Pointer to an integer variable for the data return

Description: Reads position lock time for a motor in milliseconds. Position lock time is the time allowed to elapse on completion of a move before the motor currents are dropped back to the lower holding current values. The default value of 10 milliseconds should suffice for most loads. If the motor is currently moving, this function will

	wait for it to stop before proceeding. Lock times are always in milliseconds and can range from 0 to 255ms
Example:	`nl_read_lock  15,npLockTime`
	Returns the current lock time setting in the location pointed to by npLockTime

nl_read_microsteps <mtr>,<npMicrosteps>

Command:	Read microsteps
Parameters:	<mtr> — Nanomover number, 1 through 16
	<npMicrosteps> — Pointer to an integer variable for the data return
Description:	Read the number of microsteps per motor step from the Z8 microcontroller. If the motor is currently moving, this function will wait for it to stop before proceeding.
Example:	`nl_read_microsteps 3,npMicrosteps`
	Returns the value of the microsteps per step being used by the Z8 in the location pointed to by **npMicrosteps**

nl_read_nano_ii_dio <mtr>,<upData>

Command:	Read dio data
Parameters:	<mtr> — Nanomover number, 1 through 16
	<upData> — Pointer to an unsigned variable for the data return
Description:	Reads the data from the dio port for the given motor. This function only works on Nanomotion II hardware — Nanomotion applications for earlier hardware should use the nl_read_joysticks function.
Example:	`nl_read_nano_ii_dio 3,upData`
	Returns the current dio value for motor 3 in the location pointed to by upData

Note: There is only one joystick port shared by motors 1 & 2, etc. Calling this function with either of the motors for a specific board will return the data from the same dio port.

nl_read_position <mtr>,<dpPosition>

Command:	Read motor position
Parameters:	<mtr> — Nanomover number, 1 through 16

	<dpPosition> — Pointer to an double variable for the data return
Description:	Reads the position of the motor in currently selected units. If the motor is stopped, the position is immediately returned.

FOR Z8 EPROM VERSION 1.0:

	If the motor is moving when this function is called, this function will wait for the motor to stop before reading the position and returning it.

FOR Z8 EPROM VERSION 2.0 and greater (all Nanomotion II systems):

	The routine will immediately read and return the position of the motor regardless of whether the motor is moving or not.
Example:	`nl_read_position 4,dpPosition`
	Returns the current position value in the location pointed to by **dpPosition.**

nl_read_status <mtr>,<upStatus>

Parameters:	<mtr> — Nanomover number, 1 through 16
<upStatus> -	Pointer to an unsigned integer variable for the data return
Description:	Reads the current status of the specified motor in *upStatus.
Example:	`nl_read_position 5, upStatus`
	Returns the current position value in the location pointed to by **upStatus** with the following bit values.

	Bit Number		Meaning	1	0
Low Byte:	0	=	Motor Moving	Moving	Stationary
	1	=	Left Limit Switch	Active	Inactive
	2	=	Right Limit Switch	Active	Inactive
	3	=	Emergency Stop	Active	Inactive
	4	=	Absolute Left Stop	At Absolute Left	Not at Abs. Left
	5	=	Reserved		
	6	=	At Absolute Right Stop	At Absolute Right	Not at Abs. Right
	7	=	Index Switch	Active	Inactive
High Byte:	8	=	Motor Parked	Parked	Not Parked
	9	=	Last Move Status	Move Error[1]	No Move Error
	10	=	Encoder Error[2]	Encoder Error	No Encoder Error
	11	=	Reset Interlock[3]	Reset Interlock	No Reset Detected
	12	=	Reserved		
	13	=	Reserved		
	14	=	Reserved		
	15	=	Reserved		

1A "Move Error" is declared when the position after a motor move does not equal the commanded position. ***nl_stop*** *commands, limits, encoder errors, etc. are all potential causes of move "error"s.*

2All the encoder functions in the Nanomotion II libraries require external hardware (an encoder with electrical interface) to work properly. The status bit 10 is undefined(Reserved) if no encoder is present.

3At power up, the Z8 sets a bit to one in one of its registers and initializes the motor current to 0.0. During the ***nl_init*** *procedure, the PC turns the motor current on and resets the bit to zero. If an inadvertent power reset occurs at the Z8, the current to the motor current is re-initialized to zero (preventing motion) and the bit is re-initialized to 1 which the PC can detect using this status read function.*

nl_read_units <mtr>,<cpUnits>

Command: Read unit type

Parameters: <mtr> — Nanomover number, 1 through 16

<cpUnits> — Pointer to an user-supplied buffer for the data return

Description: Copies the current units type string into a user provided string buffer.The buffer pointed to by cpUnits should be at least 10 bytes long.The following strings will represent the units:

"CM"	*Centimeters*
"MM"	*Millimeters*
"MI"	*Microns*
"NM"	*Nanometers*
"I"	*Inches*
"ML"	*Mils*
"UI"	*Microinches*

Example: `nl_read_units 5, cpUnits`

Returns the units type in the string buffer pointed to by cpUnits as one of the defined units strings.

nl_read_vel <mtr>,<dpVelocity>

Command: Read motor velocity

Parameters: <mtr> — Nanomover number, 1 through 16

	dpVelocity> — Pointer to an integer variable for the data return
Description:	Reads the velocity corresponding to nRamp (1-4) for nMotor into *dpVelocity in currently selected units (per second). If the motor is currently moving, this function will wait for it to stop before proceeding.
Example:	`nl_read_vel 15,dpVelocity`
	Returns the current velocity ramp value in the location pointed to by dpVelocity

nl_reset <mtr>

Command:	Reset motor
Parameters:	<mtr> — Nanomover number, 1 through 16
Description:	Resets nMotor to the default motion profile values as specified for the nl_init function. The units will be reset to "MM".
	Note: The position is not affected. Neither the physical position of the Nanomover nor the position registers in the Z8 microcontroller are changed by this function. The ***nl_write_position*** *function should be used if a position reset is desired.*
	The function **nl_init** calls **nl_reset** for each motor, so most programs will not need to use **nl_reset**. The **nl_reset** function is included only to provide a quick method to return the accelerations, velocities, lock times, etc. to default values. If the motor is currently moving, this function will wait for it to stop before proceeding.
Example:	`nl_reset 5`
	Returns 0 if operation is successful, non-0 if operation unsuccessful.

nl_restore_cfg <cpPathname>

Command:	Restore configuration
Parameters:	<cpPathname> — Pointer to string containing the path and filename of the file to receive the configuration data
Description:	A configuration array is used by the Nanomotion II library to contain information about the park status of a motor. The configuration array must be updated prior to

unparking the motors so that the Nanomotion II library routines know where the motors are currently parked. Generally, the configuration information should be updated into the configuration array immediately after the Nanomotion II library routines are initialized with nl_init.

The configuration information is obtained from a .CFG file and the configuration array is automatically updated when **nl_restore_cfg** is called.

The string cpPathname should be a NULL terminated path and filename that this **nl_restore_cfg** routine will use to load the information that was previously saved with the **nl_save_cfg** function. If only the filename is used, the current directory will receive the configuration file.

The MS-DOS library version of this function uses a binary data ".CFG" file. The DLL version uses an ASCII ".INI" file.

Example: `nl_restore_cfg cpPathname`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_save_cfg <cpPathname>

Command: Save configuration to file

Parameters <cpPathName> — A pointer to a string containing the path and filename of the configuration file containing the configuration data.

Description: This function writes the park information to disk in a configuration file. The string cpPathname should be a NULL terminated path and filename. The configuration file can then later be used by the **nl_restore_cfg** function to retrieve the Nanomover information when restoring the Nanomotion II system from a hardware reset or power down condition.

The MS-DOS library version of this function uses a binary data ".CFG" file. The DLL version uses an ASCII ".INI" file.

Example: `nl_save_cfg cpPathname`

Returns 0 if operation successful, non-0 if operation unsuccessful. Return Value

nl_setbase <uBaseAddress>

Command: Set system I/O base

Parameters: <uBaseAddress> — The base address for first Nanomotion board

Description: Sets the base I/O address for the Nanomotion system. If a value other than the 0x300 default for PC systems is desired, this function should be used prior to calling **nl_init**.

Example: `nl_setbase <uBaseAddress>`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_stop <mtr>

Command: Stop motor(s)

Parameters: <mtr> — Nanomover number, 1 through 16, −1 for all motors

Description: Stops one motor or all motors. An nMotor value of −1 stops all motors. If nMotor is 1 through 16, that specific motor is stopped.

Example: `nl_stop −1`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_unpark <mtr>

Command: Unpark motor(s)

Parameters: <mtr> — Nanomover number, 1 through 16, -1 for all motors

Description: If nMotor is 1 through 16, this function unparks that motor. The function unparks all motors if nMotor is -1. All park information is stored in internal variables. These variables must be initialized before this function is called (with nl_restore_cfg).

Example: nl_unpark 13

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_absleft <mtr>, <dLeftLimit>

Command: Set absolute left limit

Parameters: <mtr> -— Nanomover number, 1 through 16

<dLeftLimit> — New value for the left absolute limit

Description: Sets the absolute left limit of the Nanomover using currently selected units. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_write_absleft 3, dLeftLimit`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_absright <mtr>, <dRightLimit>

Command: Set absolute right limit

Parameters: <mtr> — Nanomover number, 1 through 16

<dRightLimit> — New value for the right absolute limit

Description: Sets the absolute right limit of the Nanomover using currently selected units. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_write_absright  5, dRightLimit`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_accel <mtr>, <ramp>, <dAcceleration>

Command: Set acceleration ramp

Parameters: <mtr> — Nanomover number, 1 through 16

<ramp> — Acceleration/velocity ramp segment 1 through 4

<dAcceleration> — New value for the acceleration

Description: Sets the acceleration ramp corresponding to nRamp (1-4) for nMotor to dAcceleration in currently selected units (per second squared). If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_write_accel  12, 3, dAcceleration`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_base_address <uBaseAddress>

Command: Set system base I/O address

Parameters: <uBaseAddress> — The base address for the first Nanomotion Board

Description: Sets the base I/O address for the Nanomotion system. If a value other than the 0x300 default for PC systems is desired, this function should be used prior to calling nl_init.

Example: `nl_write_base_address  uBaseAddress`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_bvel <mtr>,<dBaseVelocity>

Command: Set motor base velocity

Parameters: <mtr> — Nanomover number, 1 through 16

<dBaseVelocity> — New value for the base velocity

Description: Sets the base velocity for the specified motor in currently selected units. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_write_bvel  6, dBaseVelocity`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_initialized <nInitState>

Command: Set initialization flag

Parameters: <nInitState> — The desired value for the flag

1 means **nl_init** already called

0 means **nl_init** not called yet

Description: Sets the initialized value so programs can know that nl_init has been called. In general, this function should very seldom be used. The Z8 microcontroller initializes the flag to FALSE, nl_init sets it TRUE, and nl_exit sets it FALSE. The flag can be read with the nl_read_initialized function to tell at run-time whether or not nl_init has been called for the system.

Example: `nl_write_initialized nInitState`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_lmc <mtr>,<nLmcValue>

Command: Set lost motion compensation

Parameters: <mtr> — Nanomover number, 1 through 16

<nLmcValue> — New value for lost motion compensation

Description: Sets lost motion compensation in an integer range from −10 to +10. 0 disables. The units are 10 micron increments. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_write_lmc 4, nLmcValue`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_lock <mtr>,<nLockTime>

Command: Set lock time

Parameters: <mtr> — Nanomover number, 1 through 16

<nLockTime> — New value for position lock time

Description: Sets lock time (wait time) in units of milliseconds for the specified motor. Lock time is the time that the controller maintains stepping current after the motor stops moving. After the motor stops and the lock time expires, the controller drops to holding current. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_write_lock 4, nLockTime`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_microsteps <mtr>,<nMicrosteps>

Command: Set microsteps per cardinal motor step

Parameters: <mtr> -— Nanomover number, 1 through 16

<nMicrosteps> — New value for microsteps per step for the system: 25, 50, 100, 125, 250

Description: Sets the number of microsteps per cardinal motor step for the Z8 to use when moving between positions. The PC

motion control library also uses the value for calculating system resolution, etc. for maintaining units within the Nanomotion system.

The microsteps value establishes the commanded resolution for the Nanomotion II system. A Nanomover motor has 400 cardinal steps per revolution and a 0.5 mm per revolution micrometer screw. Therefore, one cardinal step is 0.5 mm / 400 = 0.00125 mm = 1.25 µm = 1,250 nm. The microsteps value is the number of discrete microsteps that this cardinal step is divided into by the Z8 microcontroller. The resolution of a Nanomover is:

Resolution = 1,250 nm / nMicrosteps.

For example, the default value of 25 corresponds to a resolution of 1,250 nm / 25 = 50 nm and is sufficient for nearly all applications. The user should remember that the mechanical resolution also depends on the construction of the Nanomover and the load being driven. With appropriate motion parameters (especially lost motion compensation), mechanical resolutions of 10nm (125 microsteps) have been achieved.

NOTE: when the microstep resolution for a z8 is changed, the current position is automatically reset to zero. the resolution change should be a power up initialization and not changed afterwards.

Example: `nl_write_microsteps 5, 125`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_position <mtr>,<dPosition>

Command: Set position

Parameters: <mtr> — Nanomover number, 1 through 16

<dPosition> — New value for the motor position

Description: Sets the current position of the Nanomover using the currently selected units. The Nanomover does not move as a result of this command — only the internal position in the Z8 microcontroller changes. Any relationship to a previously measured (as a limit switch etc.) must be maintained by the host application. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_write_position 5, dPosition`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_units <mtr>, <cpUnits>

Command: Set units

Parameters: <mtr> — Nanomover number, 1 through 16

<cpUnits> — Pointer to a string designating the desired units

Description: Sets units type for a motor using the pre-defined units string in *cpUnits. The contents pointed to by cpUnits are not changed. The following strings will represent the units:

"CM"	Centimeters
"MM"	Millimeters
"MI"	Microns
"NM"	Nanometers
"I"	Inches
"ML"	Mils
"UI"	Microinches

Example: `nl_write_units 4, "MM"`

Returns 0 if operation successful, non-0 if operation unsuccessful.

nl_write_vel <mtr>,<ramp>,<dVelocity>

Command: Set velocity

Parameters: <mtr> — Nanomover number, 1 through 16

<ramp> — Acceleration/velocity ramp number 1 through 4

<dVelocity> — New value for the designated velocity ramp

Description: Sets the velocity corresponding to nRamp (1-4) for nMotor to dVelocity in currently selected units. If the motor is currently moving, this function will wait for it to stop before proceeding.

Example: `nl_write_vel 12, 3, dVelocity`

Returns 0 if operation successful, non-0 if operation unsuccessful.

Chapter 7

Application Notes

Motorized positioners that move with the resolution and repeatability of the Nanomotion II system represent state of the art technology. There is little or no information available in the literature on how to consistently accomplish movements that are so small. Making repeatable, nanometer scale, mechanical movements on an everyday basis is still uncharted territory.

In general, positioning errors on the order of ± 1 micrometer are straightforward to achieve. But, applications that require positioning error of less than a micrometer require care and attention to every detail in the mechanical system. Because the mechanical stages and mounts usually contribute a greater error than do the Nanomovers, the need to use quality stages and components cannot be overemphasized.

The notes in this section are based on our research and development experience with motion control systems in general, and the Nanomotion II system in particular. They can help to serve as a guide to achieving nanometer size movements.

7.1 Making High Repeatability Measurements

The key to making movements with high repeatability is minimize the number of variables that are changing. It is only when a known, controlled set of operating conditions is used that accurate, repeatable movements be made with any degree of consistency.

The most important of these operating conditions is the temperature of the Nanomover, which is affected by both the ambient temperature and the heat generated by the Nanomover itself. Minimizing the effects of ambient temperature is discussed in **Dealing with Temperature Variations**. Heat generated by the Nanomover can be considered and its effects reduced with careful consideration.

The most obvious source of heat is the motor. Because of the difference between stepping and holding currents, the motor will generate more heat when moving than when stationary. Further, the motor will generate different amounts of heat depending on the velocity at which it is moving. A small portion of the motor heat will be conducted through the Nanomover housing to the leadscrew, causing positional errors.

Another important, but less obvious, source of heat is the friction between the nut and the leadscrew. The amount of heat generated is a function of speed and load. This nut/leadscrew heat and its resulting temperature change will directly change the length of the leadscrew.

The amount of heat generated by Nanomover motion is difficult to determine. But, the effects of motor heat and nut/leadscrew friction can be minimized by simply letting the heat dissipate. Waiting approximately 20 seconds between quick short moves (1 second of movement or less) will allow the Nanomover to dissipate the extra heat and reach a constant thermal point in which the effects of motor heat and nut/leadscrew friction are negligible. For longer movements it may be necessary to wait for a longer period of time.

Note that both of these internal heating effects are quite small in absolute terms, perhaps producing 250nm of error under typical conditions.

Lost Motion Compensation keeps the positioners themselves constant, and will even remove many lost motion errors in the stages used.

Another important, but often neglected, source of error that can be easily removed is strain relief in the Nanomover cables. The cables should be strain relieved away from the Nanomover so that any disturbances in the cable will not translate to measurement or positioning errors. The cables may be strain relieved by fixing them to a solid surface.

7.2 Dealing with Temperature Variations

Thermal expansion due to slight temperature variations can significantly affect the resolution of high-resolution mechanical systems. It is physically impossible to make highly repeatable moves if the temperature changes as little as a degree. It is therefore important to make every effort to maintain the temperature of the system components at a constant level.

The performance of the Nanomovers is specified for 20 degrees C. Often, it will be necessary to maintain the temperature and operate the system at some other value. It is still possible to make highly repeatable movements at other temperatures, provided the operating temperature is held constant. While the absolute position of the Nanomover will be slightly displaced, a displacement correction can be calculated using the thermal expansion coefficients described in Installation and Setup.

In addition to controlling the ambient temperature as much as possible, the Nanomovers must be shielded from drafts and wind currents caused by air conditioners, heaters, and similar equipment. Do not place the Nanomovers in an environment where large temperature fluctuations can occur quickly.

Mounting the test or experiment on an optical table is desirable not only from the mechanical standpoint, but from the temperature standpoint as well. The fairly large thermal mass of the table will help keep the Nanomover temperature at a constant value.

Before doing any experiments or tests with the system, it should be allowed to temperature stabilize for at least one hour. To temperature stabilize, a Nanomover must be unparked or moved and then maintained in that position, because current is present in a Nanomover only after it has been unparked or moved.

(This page intensionally left blank.)

Appendix

A-1 Certification and Repeatability Procedure

The performance of each Nanomover is tested and certified with a Hewlett-Packard laser interferometer (model 5528A) before it leaves the factory. Each Nanomover is certified to have an accuracy of ±1 micron and a bidirectional repeatability of better than ±100 nanometers. Included with each unit is a certification graphs showing its accuracy and repeatability over the full travel range.

A test program directs the system through a standard test course, which is monitored by the laser interferometer. Results from the Nanomotion II system and the interferometer are then used to calculate accuracy and repeatability. The following algorithm is used to generate the accuracy and repeatability graphs.

For each 1 mm in position along the travel line:

- After approaching the current position with a positive move, read the current actual position (location 1).
- Move 1 mm in the positive direction.
- Move 1 mm in the negative direction.
- Read the current actual position (location 2).
- Calculate the bidirectional error ((location 1)-(location 2)) and plot it.
- Move to the next position 1 mm down the travel line, and repeat the process.

LMC is set to +2 for all testing. The weight of the system being moved is a constant 1.5 kilograms. The temperature of the screw is measured and recorded at the end of the test, and is printed on the certification graph.

Interpreting the Nanomover Certification Graph

Each Nanomover is certified to have an accuracy of ±1 micrometer and a bidirectional repeatability of better than ±100 nanometers. The figure shown below is a typical certification graph, with the upper graph a plot of the absolute accuracy, and the lower graph a plot of the bidirectional repeatability error. The sample interval of both graphs is 1 mm.

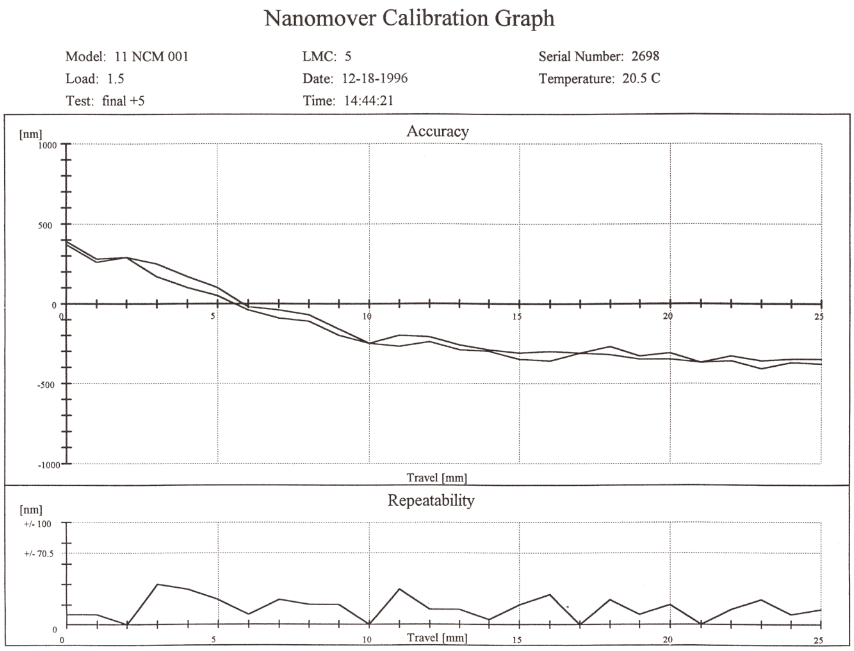


Figure A.1 Motor Calibration Graph

The bidirectional repeatability error is obtained by plotting the error obtained by taking the absolute magnitude of half the difference in actual positions for the same location, when approached from the positive and negative. The difference is divided by two because repeatability is conventionally defined as a symmetric error.The highest peak on this graph, at 50 nanometers, therefore represents a worst case bidirectional repeatability of 50 nanometers. A typical bidirectional repeatability error is around $\pm$20 or 30 nanometers even though the guaranteed performance is $\pm$100 nanometers.

The absolute accuracy is obtained by plotting the Nanomover's position as determined by the interferometer vs. the its expected position.

Sources of Error

Nanomotion II has three main sources of systematic, reproducible errors in the mechanical system, which contribute to the accuracy error.These are the motor tooth pitch error, torque errors, and the leadscrew error.

The motor tooth pitch error is a deviation of the tooth to tooth spacing in the motor. It causes the actual microstep size to decrease or increase linearly between teeth as a function of the tooth pitch error. The manufacturer of the motor certifies that the tooth pitch error is within ±3%, and typically is ±1%. This translates to a positional error of 150 nanometers maximum, typically 50 nanometers, and is generally not a concern. The error is non-cumulative beyond one motor revolution.

Torque errors will cause the microstep size to increase or decrease in a sinusoidal manner within a tooth pitch. These errors are caused by non-sinusoidal torque characteristics of the motor, and variations in applied energy due to imperfect electronics. The Nanomotion II has torque compensation that minimizes the effects of torque error. The total torque error is typically less than 50 nanometers and is generally not a concern. This error is non-cumulative beyond one full (cardinal) motor step.

The lead screw error is the main contributor to the overall accuracy error. The accuracy certification graph is primarily a measure of the lead screw error. This error is typically in the range of ±500 nanometers and should be considered when making absolute measurements. Software modifications can remove much of this error, if it is a significant drawback in a particular application.

Performance of Systems with Stages

It is possible to purchase the Nanomotion II system with the motors already attached to stages (see the Melles Griot catalog for current part numbers and options). There have not been any specifications compiled for the performance (accuracy) of the motors and stages as a system.

A-2 Setting Acceleration and Velocity

Movements of the 11 NCM 001 and 11 NCM 007 Nanomover motors can be increased or decreased using the acceleration ramps discussed in Chapter 2, Introduction to Nanomotion II. Figure A.2 shows typical limiting acceleration versus velocity for various inertial loads. To generate optimum velocity, the first step is to determine the maximum load the Nanomover will have to drive. Typically this would be approximately given by the springs on the stage used, unless an untypical load is being mounted to the stage.

Next, locate a load line on the graph that most closely approximates your worst-case load. If the maximum load falls between two lines, a new line can be approximated by interpolating between the load lines. Locate three roughly equidistant points between the top left corner of the curve and the

desired slew rate. These points represent good choices for acceleration values from the graph. Enter these values for the ramp segments. Use an acceleration value ≥1200 mm/sec^2 in ramp #1 to enter the graph through the top left corner. The maximum achievable value will depend on the application.

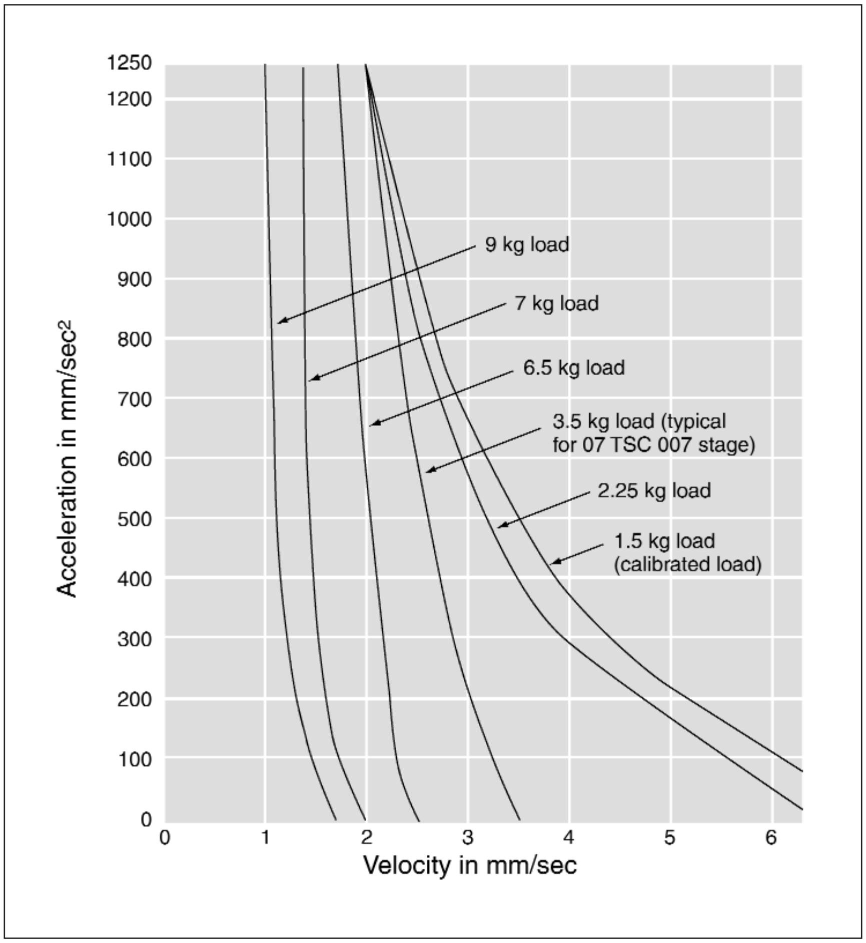


Figure A.2 Standard curves for motor velocity vs acceleration

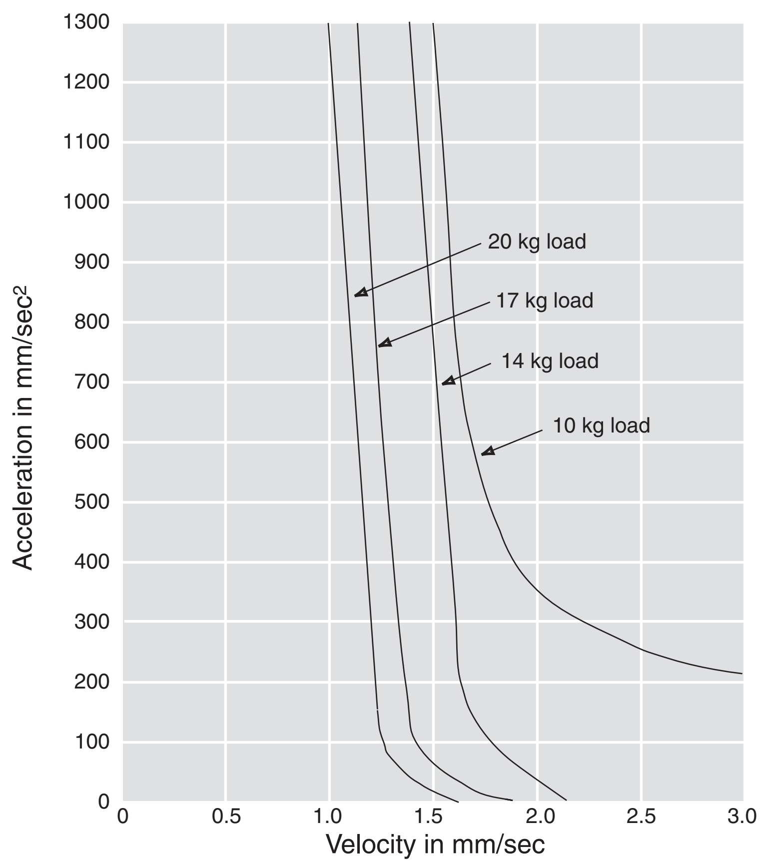


Figure A.3 Velocity vs acceleration for11 NCM 005 high-torque motors

Index

NANOMOTION™ II USER MANUAL

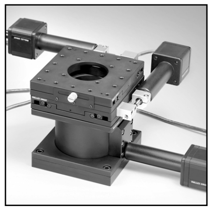
Controller Model Number

- ❏ 11 NCS 101/IBM
- ❏ 11 NCS 101/IEEE
- ❏ 11 NCS 101 Expansion Module

Serial Number: ____________________

Purchased by: ____________________

Date: ____________________



the practical application of light

MELLES GRIOT

Photonics Components
16542 Millikan Avenue, Irvine, CA 92606 • 1-800-835-2626 • (949) 261-5600 • FAX (949) 261-7790 • E-mail: sales@irvine.mellesgriot.com

Asia +81 (03) 3407-3614 **Europe** +31 (0316) 333041

www.mellesgriot.com

A member of Barlow Scientific Group Ltd.